QL Today

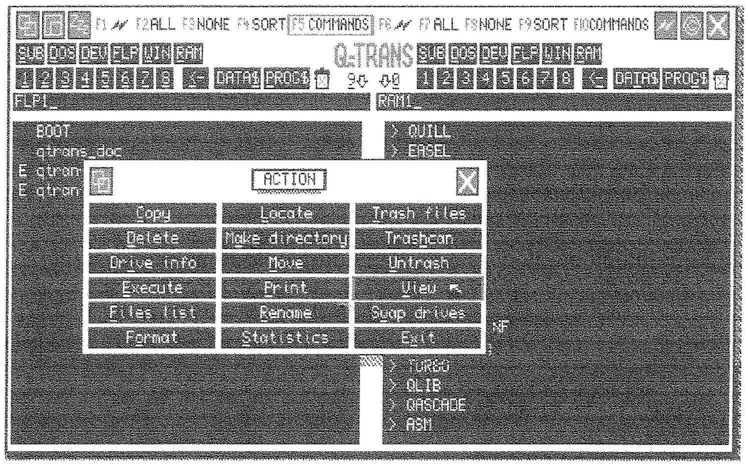
Volume 7
Issue 6
March/April
2003

ISSN 1432-5454

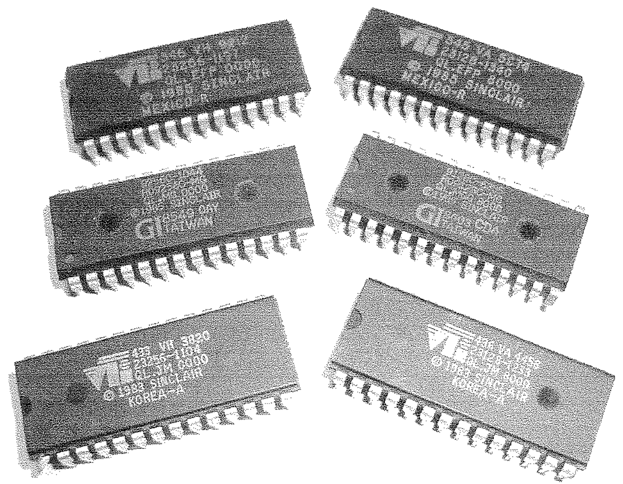
The Magazine about QL, QDOS, Sinclair Computers, SMSQ...

3D QD in Style Window Manager

Q-TRANS



REVIEW



QL ROMs

another look at them and their successors

Contents

Advertisers

in alphabetical order

QL *Today*

ISSN 1432-5454

German office & Publisher:

Jochen Merz Software	Tel. +49 203 502011
Im stillen Winkel 12	Fax +49 203 502012
47169 Duisburg	BBS +49 203 502014
Germany	email: JMerz@j-m-s.com
	email: QLToday@j-m-s.com

English office:

Q Branch	Tel. +44 1273 386030
20 Locks Hill	Mobile +44 7836 745501
Portslade	Fax +44 1273 381577
BN41 2LB	email: qbranch@qbranch.demon.co.uk
United Kingdom	email: QLToday@j-m-s.com

Editor:

Dilwyn Jones	Tel. +44 1248 354023
41 Bro Emrys	Fax +44 1248 354023
Tal-Y-Bont, Bangor	email: dilwyn.jones@dj.softnet.co.uk
Gwynedd	email: QLToday@J-M-S.com
United Kingdom LL57 3YT	

Co-Editor:

Bruce Nicholls	Tel +44 1708 510764
38 Derham Gardens	Fax +44 870 0568755
Upminster	email: qltoday@q-v-d.demon.co.uk
Essex RM14 3HA	email: QLToday@j-m-s.com
United Kingdom	

QL *Today* is published bi-monthly, our volume begins on beginning of June. Subscriptions begin with the current issue at the time of sign up. Please contact the German or English office for current subscription rates.

We welcome your comments, suggestions and articles. YOU make **QL *Today*** possible. We are constantly changing and adjusting to meet your needs and requirements. Articles for publication should be on a 3.5" disk (DD or HD) or sent via Email or into one of the JMS-BBS's. We prefer ASCII, Quill or text87 format. Pictures may be in _SCR format, we can also handle GIF or TIF or JPG. To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hardcopy of all screens to be included. Don't forget to specify where in the text you would like the screen placed.

Article and Advertising deadlines are as follows:
Issue 1: 30 April
Issue 2: 30 June
Issue 3: 30 August
Issue 4: 30 October
Issue 5: 30 December
Issue 6: 28 February

Dilwyn Jones

I was writing in the editorial column in the last issue that it might have sounded like I was going out of my way to be positive. I made no defence for this, indeed I listed several reasons why I was being positive.

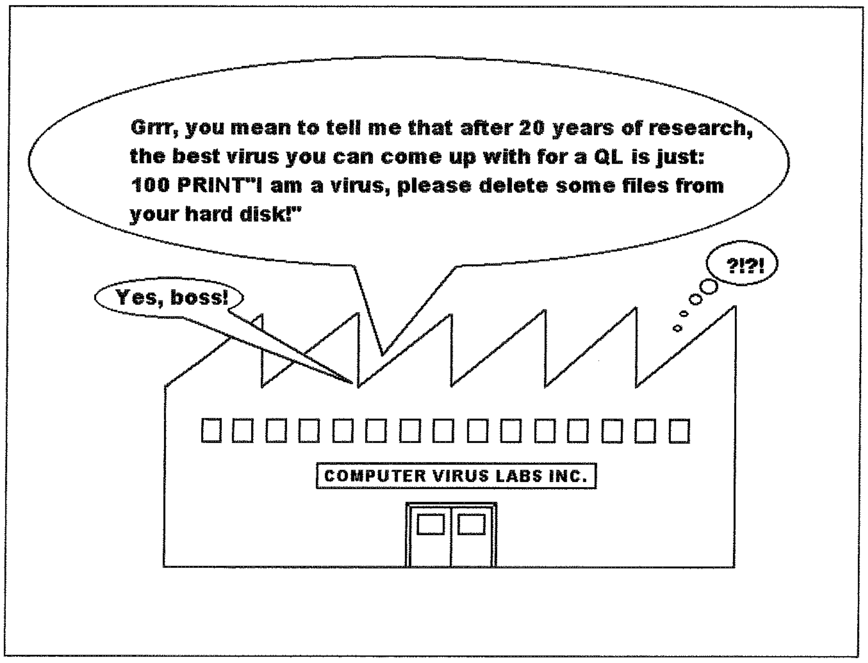
A few people contacted me after reading it to say how much they appreciated the positive viewpoint and indeed confirming the information given in recent news pages. Their views seemed to be that it was all too easy to get negative or to allow one or two negative views to distort the overall picture of a QL scene which is still lively and healthy, though fairly small, after 20 years.

Shortly after that issue of QL Today appeared, Jon Dent sent an email to the ql-users email mailing list to confirm that soql PPP was working - the email was sent using his system - so we know that soql is one huge step nearer completion, the holy grail for QLers of email and internet access is now that one step nearer! And with a lot of work taking place on the operating system (colour drivers, window manager and GUIs) and new programs appearing regularly, we have every reason to be quietly proud of the QL scene in 2003!

I was very glad to learn that what little feedback I got on last issue's cover disk was also very positive. It was a bold move by the publisher admittedly, issuing a cover CD-ROM not really knowing how many readers would be able to use it, and it seems to have paid off, as I had a number of messages thanking us both for the CD itself and the subject matter contained on it, so we seemed to have a success on our hands!

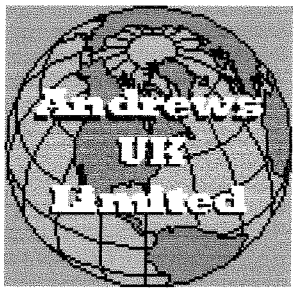
This issue will consist of a lot of listings, so plenty of typing practice for you. By the time of publication or shortly afterwards, I hope to have put the listings onto my website or the QL Today website (or both!) for those who prefer not to type them in. Either way, I very much hope you will enjoy the articles, as we bring to a close our seventh year of publication! I wonder where Volume 8 of QL Today will take us?

Soon, it will have been 20 glorious years of the QL. Why don't we all pause and think how to celebrate 20 years of our favourite computer and operating system. Let us know your views!


Grrr, you mean to tell me that after 20 years of research, the best virus you can come up with for a QL is just: 100 PRINT"I am a virus, please delete some files from your hard disk!"
Yes, boss!
?!?!
COMPUTER VIRUS LABS INC.

NEWS

A company called Sinclair Computers have contacted us to say that they have a website which specialises in Sinclair computers, covering everything from MK14 to ZX80 through QL, Z88 and the PC200-500 systems and that they have for sale a large amount of second-user computers and accessories for sale. The site includes a lot of information, such as access to lots of historical information on the older computers and a more general retro-computing site as well. Their website is at the intriguingly named
www.sinclaircomputers.com
address (they emphasise that they have no direct connection with Sinclair Research at all). For those familiar with the Quantum Ring linked list of QL related websites, this site appears to be a member of that system, making it easy to navigate from one QL site to another!
The site is owned by Andrews UK Limited.
For more information, email Paul via email at **info@sinclaircomputers.com** or by letter at:
Sinclair Computers
P.O. Box 2243
Leagrave
Luton
Bedfordshire
LU4 0YR
England



Q-TRANS

v1.03 of Q-Trans file handling program is now available. The latest version fixes a serious bug in the configuration block system, which could corrupt the configuration and parts of the program if string items in the configuration block were set to longer than the default settings supplied in the program.
http://homepages.tesco.net/dilwyn.jones/software/freeware.freeware.html

QemuLator runs SMSQ/E!

After the recent news of the arrival of SMSQ/E v3, the colour drivers for Aurora and the new Window Manager with SMSQ/E+GD2 systems comes an announcement that the latest version of QemuLator for PCs can run a version of SMSQ/E designed for QL+Gold Card systems. While this does not in itself give QemuLator GD2 or high colour support, it does at least offer some of the benefits of the SMSQ/E version of our favourite computer's operating system. Al Boehm reports in the February 2003 edition of Quanta newsletter that v2.3a2 of QemuLator was successfully running SMSQ-Gold v2.98. This was, however, an early test version of the emulator, developed following the release of the SMSQ/E sources last year, which allowed Daniele Terdina, author of QemuLator, to make developments ot the emulator to allow it to LRESPR the Gold Card version of SMSQ/E. More news as we get it! The official QemuLator website is at:
http://users.infoconex.com/daniele/q-emulator.html

Geoff Wick's New Email Address

Please note my new email address:
gwicks@beeb.net
This is the prefered address to use from now on, although I shall keep the hotmail address for a little while yet. I'd miss all the spam if I didn't have it!

DISPLAY_CODE Extensions

The display_cde extensions originally written for QL Today (see Vol 2 issue 3 September 1997) have now been updated to add five new BASIC functions:

`GD2` returns 1 if "colour drivers" present, 0 if not

`PTR_ENV` returns 1 if pointer interface present, 0 if not

`WIN_MAN` returns 1 if window manager available, 0 if not

`PTRVER$` returns pointer interface version number

`OS_VER$` returns the QDOS or SMSQ operating system numbers

`WMAVER$` returns the Window Manager version number

These new functions, written with much appreciated advice from Marcel Kilgus, were written to let programs extract information from the system to see if required facilities are likely to be present. For example, a program which needs colour drivers can test if high colour drivers are there, and if not, stop gracefully with a simple command like:

```
IF GD2(#0)=0 THEN PRINT "Sorry, I need
GD2!":STOP
```

While SBASIC already provides facilities to extract most of this information from the system, these extensions work in both SuperBASIC and SBASIC and can be used with Turbo and QLiberator compiled programs too. They are freeware

and may be used in PD or commercial programs if you wish.
I have left the old version available for now, in case someone finds bugs in the new version!
The new versions are available in a file called Display2.zip available for download from the My Freeware page on my website:
http://homepages.tesco.net/dilwyn.jones/software/freeware/freeware.html

News from George Gwilt

1. GWASS v 4e19
A beta test version of GWASS is available on the SQLUG site. This version allows the use of expressions for numbers. Expressions can involve brackets and labels as well as numbers. Also precedence of operators is recognised so that
```
    1 + 2*3
```
is taken as
```
    1 + (2*3)
```
Choice can be made between the old and new methods of inputting numbers.

2. TurboPTR
Two new extras have been added to TPTR_EXT v 3.7. They are BSVPW, which saves part of a window into a buffer, and BRSPW which restores an area of a buffer to a window.
The format of the buffer is that described in the QPTR Manual.

3. Turbo
The possibility of allowing larger extension files in Turbo is being investigated. The current size limit is just under 32K.

Gwass, TurboPTR and Turbo compiler may be downloaded from the SQLUG website, the address of which is
http://www.jms1.supanet.com

QUILL, ABACUS, EASEL and ARCHIVE V1

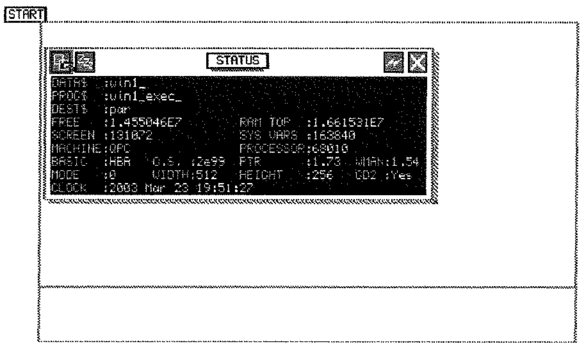
Collectors of older QL software may like to know that I have made available copies of the original version 1 of the Psion titles for the QL. Whilst these are not particularly attractive from a usability point of view (version 1 was quickly superceded by the more compact and more reliable version 2), they may be of interest to collectors and retro enthusiasts. They are available from my PD Library service and from the Psions page on my website. The same sources also contain copies of the more recent versions 2.0 onward, and some international versions, which may be useful to users whose copies have become corrupted over the years, or those who may have purchased a second-user QL which came without copies of the original software.
http://homepages.tesco.net/dilwyn.jones/psions/psions.html

Darren Branagh

We all know Darren Branagh as the jolly Irish face of Q-Celt Computing. Well, as of 18th March 2003 there are now two Darren Branaghs in this world! Yes, his girlfriend Sandra gave birth to son Dylan Jack Branagh, born at 10.57am. Weight: 3.103 kg. Length: 56cms. Darren says: "I'm over the moon!" I'm sure we'll all join in sending Darren and Sandra our congratulations and best wishes. Rumour has it that Dylan Jack has already demanded a QL of his own...

Status Program

I have released a small program I call STATUS which is basically an on-screen display of machine status information, anything from DATA/PROG/DEST default settings, to clock, operating system and pointer environment version displays, presence (or not) of colour drivers, screen details and machine and processor information. Suitable for pointered and non-pointered systems, but of greatest use on more recent systems. It may be downloaded from the My Freeware page on my website, and in time available from most PD libraries.
http://homepages.tesco.net/dilwyn.jones/software/freeware/freeware.html



```
DATA$  :win1_
PROG$  :win1_exec_
DEST$  :par
FREE   :1.455046E7           RAM TOP  :1.661531E7
SCREEN :131072               SYS VARS :163840
MACHINE:QPC                  PROCESSOR:68010
BASIC  :HBA    O.S. :2e99    PTR      :1.73   WMAN:1.54
MODE   :0      WIDTH:512     HEIGHT   :256    DD2 :Yes
CLOCK  :2003 Mar 23 19:51:27
```
Screen dump of Status program

SOQL News

On the 9th March, Jon Dent announced that he has a working PPP implementation of the soql TCP/IP stack system for the QL. He sent a test email to the ql-users mailing list to prove it works, and announced he is looking for volunteers to try out soql PPP in various countries. You need to find an Internet-by-call provider of which there seem to be plenty but he can only test Swiss ones from where he is based. Once connected you can use your usual email accounts if you can enter their URLs with hexadecimal equivalent in the TCP_DNSrecords_txt file. "You need to have a bit of time to play about with it as there is no smooth human interface or automatic installation. but if you can read this it must be working :-)" was what Jon said. It is obviously very good to hear that such progress is being made with this software, which is keenly expected by so many QLers.

ERGON Development News

Davide Santachiara writes:

I have updated MasterBasic and DEA so that they should be now GD2 compatible. I tested the new routines on MODE 32 only (QPC2 v3.03) so I would welcome if somebody could test other modes (ie. 16/256 colours modes).
They are available as usual from my web site
www.geocities.com/dsantachiara

NESQLUG votes to become NASQLUG

Al Boehm writes:

In the January Virtual Meeting conducted by email, New England Sinclair QL User Group (NESQLUG) voted to change its name to North American Sinclair User Group (NASQLUG). For some time the majority of members have been located outside the New England area.
All officers, rules, and traditions remain the same.
If you would like to join NASQLUG, contact the treasurer Kevin O'Leary
email: **olearyk@cyberportal.net**
or query Al Boehm
email: **albertboehm@juno.com**
for more information.

Beginners Club Italy

Andrea Carpi writes:

Since today the pages dedicated to Sinclair QL edited by QL user group of Beginners' Club, completely renewed, have the following new address:
http://ql.beginnersclub.org
I invite whoever has some links with the pages of BC to update his web site.
The old address
http://www.beginnersclub.org
remain for other activity of the Club, not inherent to the Sinclair QL.

Good News on QL and Spectrum Membranes

Seltech of Germany has made a big investment supported by Rich Mellor, and Bill Richardson to get an initial 500 batch of QL Keyboard Membranes made, and they should be available in April – ZX Spectrum keyboard membranes are available now.
Rich Mellor (aka RWAP Services) will be selling them, and in order to recover the advance payments which were necessary to get them made the basic cost will be £17.50 plus post and packing. However, to encourage early orders,any placed before 31st March 2003, and paid in advance will be sold for £15.50 plus post and packing. We hope QL users who still use membranes will support this initiative with their early orders.
Post and Packing for 1-3 membranes – please ask for larger amounts: UK : £2.50 EEC: £4 Rest of Europe: £5 US/Canada: £5.50
Rich Mellor RWAP Software
35 Chantry Croft, Kinsley, Pontefract,
West Yorkshire, WF9 5JH TEL: 01977 610509
http://hometown.aol.co.uk/rwapsoftware

We wonder why so few of our readers take advantage of placing small ads in QL Today. There are still QL users out there looking for old QL hardware - even things you are going to throw away. I had several inquiries about QL power supplies, for example.
So, if you want to get rid of QL-related items, why not advertise them here - it's free, after all and you could make somebody else happy.
Please note that the deadlines given on page 2 apply for small ads as well - but you can always send them in. If they miss the forthcoming issue, they'll get into the next one.

Small Ads

For Sale

I have acquired a quantity of original EPSON inkjet cartridges for the Epson 850 printer. Some are slightly out of date, but otherwise boxed and in good condition – should work fine.
Black cartridges £7.50 ea.
Colour cartridges £9 ea.
Post and packing extra.
RWAP Software - Rich Mellor (see above)

QD in 3D Style WMAN

Jochen Merz

It has been done: QD has been converted to reflect the new 3D look by using the hi-colour driver and new Window Manager.

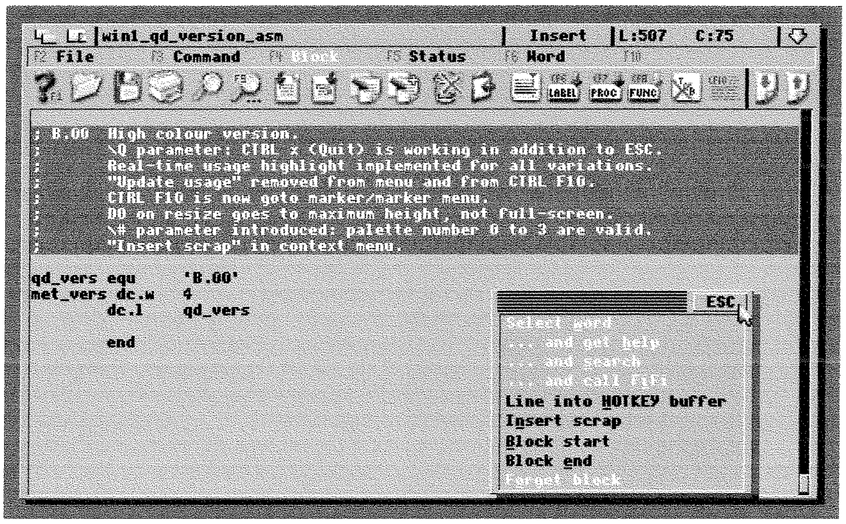
First, I would like to thank Marcel Kilgus for implementing this great new Window Manager, thanks to Phoebus Dokos for providing the wonderful looking sprites and thanks to Bernd Reinhardt for having started the conversion on QD ... the result looks impressive. Bernd and myself have added the final touches to QD, QSpread etc. during the evenings of our skiing holiday (more or less the only time where I actually find the time for programming) and here is the result.

Not only has the whole look been changed (well, have a look at an "old" looking QD next to it and you'll see the difference) but a few things have been added too.

First, the line highlights depending on the usage have been properly implemented. This means, the highlight colour always reflects the **correct** state of every line at any time. No need to use the CTRL F10 to refresh the colours, no odd coloured lines anymore. This applies to all usage settings: once you create or modify a line and you move the cursor away from it, the highlight is added or removed, depending on the contents of the line. A great feature, now much more useful.

Well, and as the CTRL F10 icon in the toolbar was without any function left, we thought ... well, how about placing the "marker" functions there - much easier to be accessed than from the menu. We placed it there and noticed after a day that now, since it is much quicker accessible, we actually use it more often.

We have also modified the behaviour of QD when you "DO" the resize icon. Previously, it went to full-size to occupy the full screen. Not very useful when you're running several QDs in 1024x768 resolution. We found when we talked to our customers about this behaviour, that bringing the height of QD to full size only would be so much more useful, so that's how QD reacts now. And it *is* so much more useful.



```
; B.00  High colour version.
;       \Q parameter: CTRL x (Quit) is working in addition to ESC.
;       Real-time usage highlight implemented for all variations.
;       "Update usage" removed from menu and from CTRL F10.
;       CTRL F10 is now goto marker/marker menu.
;       DO on resize goes to maximum height, not full-screen.
;       \# parameter introduced: palette number 0 to 3 are valid.
;       "Insert scrap" in context menu.

qd_vers equ      'B.00'
met_vers dc.w    4
        dc.l     qd_vers

        end
```

We have also placed the function "Insert scrap" into the context menu. That was missing here, as a number of QD users told us.

The new Window Manager allows the user to define up to four totally different colour schemes (we use siver/grey and golden - both look very good). You can, of course, configure QD to use the desired screen on startup, and you can, of course, pass a key in the command string to QD on startup to have differently looking QDs for dedicated jobs (e.g. QDs for BASIC programming in one colour scheme, QDs for text editing or assembler programming in a different colour scheme). Very useful, that.

You will probably wonder about upgrading QD. Well, by the time I write this text I don't know when the new SMSQ/E will be released - and we can't release a new QD until SMSQ/E and the new Window Manager are available. It is possible that the Registrar releases it at the time you read this, and if there is some last-minute news, it will be reflected in the J-M-S advert somewhere in this issue. Everything is working perfectly well as it is right now, but there is still some fine-tuning in SMSQ/E going on (i.e. which sprites become system sprites to reduce program size) It will definitely be an upgrade - not free, but not expensive. You'll get a new QD (we need to call it QD 2003, don't we? - although QD '03 sounds better if you pronounce it), a new Menu-Config and, of course, a new Menu Extension - all featuring the new 3D style look.

QSpread '03 or 2003 will be available at the same time - it is ready too, and we need to wait for the new SMSQ/E as well - same situation as with QD.

One thing is for sure: once you're used to the new colours, you don't want the old look!

QL ROMs

David Denham

I was flicking back through old copies of QL World magazine recently and read through some very interesting articles by Simon Goodwin about QL ROMs. His articles mainly referred to the problems he had encountered and documented in the ROMs issued by Sinclair Research for the QL. There was also a three part article by Mark Knight in QL Today on a similar theme.

While browsing a PD software library catalogue recently I noticed that copies of some QL ROMs are now available on disk. As I understand it, it is now possible to copy QL ROMs within Britain and Europe at least, although you need to get permission to copy or distribute them in America, as the rights to some QL products are still held by two individuals in the United States (Paul Holmgren and F.W. Davis).

I thought that rather than look in detail at the various ROM bugs and other such articles already written by Mr. Goodwin, I'd take a rather more simplistic and general look at the various ROM versions to try to impart some information as to what the differences are, which is oldest and which is newest and so on.

I will generally refer to the ROMs by their version of SuperBASIC, which of course we would normally find with the command PRINT VER$. This returns a two, three or four letter identifier. Many theories abounded in the early days of the QL as to the significance of these letters, the most common theories being that they were named after the initials of Sinclair staff or taxi drivers used by the company at the time.

There is also a version number for the operating system itself. This is usually a 4-digit number such as 1.03 and this refers to QDOS or SMSQ version number rather than SuperBASIC or SBASIC. There is no facility in original QL ROMs to check this version number from BASIC, although there is a facility which can be called from machine code to check it, and Andrew Pennell among others wrote an extension called QDOS$ to check this ROM version - the listing was published in his book The Sinclair QDOS Companion (page 130) published by Sunshine Books in 1985. Although the book is now out of print, copies still turn up from second hand QL equipment suppliers and at computer shows from time to time. Many BASIC extensions toolkits have similar extensions to check the operating system version. If your system has Minerva or SMSQ then you can use a special version of VER$ to check this operating system version:

```
PRINT VER$(1)
```

QLUDGE

A horrible word, but it was used to refer to the little board which plugged into the back of the QL carrying early QL ROMs (see picture in fig. 1). This carried an EPROM version of the operating system which, if I remember right as it was nearly 20 years ago, was called version FB and I think QDOS version 1.00. Whatever the version, it was horrible. There were innumerable ways of crashing the QL and needless to say, it was quickly replaced by improved versions. Version PM used QDOS 1.01 and was a slight improvement, but still less than ideal.

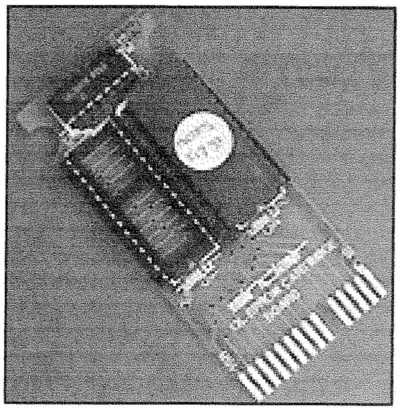


Figure 1 - the "Qludge" EPROM board

AH ROM

This was perhaps the first seriously useable versions of the QL ROM. It was fairly stable, in widespread use (many QLs still exist with this version). It used QDOS version 1.02. Like the next version (JM) it lacked the error trapping provided by WHEN ERROR, REPORT, ERLIN etc commands and a few other facilities introduced in later ROM versions, but it is perfectly possible to use the AH ROM today.

JM ROM

Often rumoured to be named after John Mathieson, an engineer who worked for Sinclair at the time, this ROM was quite close to version AH in many ways, with various slight improvements in QDOS 1.03. Of the early ROM versions, this was probably the most widely used and is a reasonably good version of the ROM. It is stable in use and the bugs are well documented. I know a few software authors who have said that they prefer writing software on a version JM QL because they know that in general if it will run on a version JM QL it will work on anything, one the basis that this was the last release before the more common JS ROM which is still in use today.

TB ROM

I know little about this ROM except to mention that it contains QDOS 1.03 like the JM ROM, and it seems to be an intermediate release somewhere between JM and JS. As far as I know, it is not in widespread use.

JS ROM

This contained QDOS version 1.10 and introduced the new error trapping keywords and other facilities for BASIC programmers. These were working but not complete - a few problems became obvious after a while. Despite this, the JS ROM went on to become the most widely used version of the QL ROM. A special version (called JSU) was released for the launch of the QL in the USA, but as far as I know, this was the only international version of this ROM.

JS ROM VARIANTS

The main variation on the JS ROM was by a company called Thor who produced a QL compatible computer in the mid 1980s. This computer used a derivative of QDOS called Argos, which according to Simon Goodwin gave QDOS versions of 4 and 5 but were really quite close to the QDOS of JS ROMs. A more recent variation of JS is called JS4M, and it's really a version of the JS ROM patched by QL emulator authors to allow use of up to 4MB of RAM, although it still identifies itself as QDOS 1.10.

MG ROM

The MG ROM was designed to be an improvement on JS and to provide national versions. MG ROMs contain QDOS version 1.13. The country version was coded into the version name of the ROM as a third letter (MGF for France, MGE for Spain, MGD for Germany, MGI for Italy and so on - I don't know how many national variations existed) and in the QDOS version number, the '.' was replaced by the country identifier letter, so the French MG ROM would have been 1F13 for example. The MG ROM was widely used in Europe, but the JM and JS ROMs seem to have been the most widely used versions in Britain.

MG VARIANTS

Some non-Sinclair versions of the MG ROM seem to have been unofficially produced over the years. A version called MGUK was released by John Alexander. This contained fixes for known bugs in QDOS at the time and introduced some new commands into the bargain. Although available from some software traders at the time it did not seem to get into widespread use, whether that was because people did not trust unofficial releases or for copyright issues worries I am not too sure.

A version MF with QDOS 1.14 exists, and it seems to be designed for use on German QL systems, so it seems reasonable to assume it was produced either for German-speaking areas or by a private individual in one of these countries. I know little about it other than having a copy on the PD library disk I mentioned.

Another version of QDOS 1.14 was contained in the Ultrasoft version of the MG ROM. Again, I know little about this other than the fact that it seems to be intended for use with German QLs.

TYCHE ROM

This is a 64K ROM with QDOS version 2.05 and contains the copyright string 'Copyright 8519 Siris Cybernetics', so despite the two character reversal was probably released in 1985 if it was ever actually released. If I have understood Simon Goodwin's articles properly, the name 'Tyche' seems to mean 'fate', so if this was to have been for a QL successor machine, the name must have seemed ironic in view of what happened to the QL during the time of the Amstrad tie-up.

MINERVA

This is probably the best known of the non-Sinclair ROMs. It was originally produced by a small team known as the QView Mega Corporation. QView was comprised of Stuart McKnight, Jonathan Oakley and Laurence Reeves. When QView ended, Laurence Reeves continued with the development of Minerva and it was marketed by Tony Firshman at TF Services. Minerva gives a VER$ of JSL1 (Jonathan, Stuart and Laurence appropriately enough and the QDOS version varies depending on which of the large number of releases we are talking about. Minerva righted many of the bugs in earlier Sinclair ROMs and added many new facilities and improvements such as faster graphics. Over time, the Minerva version of the QL ROM became so different and distinct to standard QDOS that it could not be seen as a copyright infringement in any way, it was a true alternative to Sinclair QDOS. The name Minerva comes from Roman mythology - Minerva was the goddess of wisdom, the invention and arts, a well chosen name perhaps!

TF Services have been reported as moving towards a more open source approach to Minerva where it may either become freely distributable or sources made available. In the meantime, one particular version of Minerva, version 1.89, has been made freely copyable for use with QL emulators where copyright issues prevent use of Sinclair ROMs.

SMSQ

This is the best known of the compatible operating systems, but is not QDOS as such. It runs almost all programs which QDOS ran, has a whole host of extra facilities like built in pointer environment, ability to read DOS disks, much faster and enhanced BASIC, modularity allowing easy extension - and as we've seen recently a degree of openness in development. You can get the sources from the registrar if you wish to see how it works or participate in its future development.

References:

The ROM images are available on a two disk set called QL07 from Dilwyn Jones's PD Software Library. I am told they are also available on his website for download where copyright issues permit.

Simon Goodwin's articles referred to were published in QL World issues dated August 1987, September 1987, June 1988, February 1989 and December 1990.

Mark Knight's articles were published in QL Today Volume 3 issues 3-5 from September 1998.

News about Suqcess

Wolfgang Uhlig

For quite a while there has not been any news about **Suqcess**, the pointer-driven database-frontend. This was, I have to admit, due to my personal situation in which there was not much room for the QL. Fortunately Bob Spelten from The Netherlands, who had been a user of my program from the very beginning and who had given me some ideas for improvements, was glad when I gave him the source code of **Suqcess**, telling him he was free to improve it himself. And that's what he did! In the meantime his latest version is 1.15 (where my last version was 1.12) and he has done a very good job, I must say. Not only did he eliminate minor bugs (even a workaround for one of DBAS's faults), but several improvements were made to the program itself. Here are some of them:

- The import is less problematic now. You even have the possibility to choose an own delimiter when importing text-files.
- Fields can be aligned left, right or centered separately and a favorite alignment can be saved for the future
- Floats can have fixed decimal places
- Duplicate has now it's own key and icon and is therefore much easier available
- The search criteria of all the named searches can be shown
- The HELP-file has been extended so that you can jump straight to the right chapter from the help menu. Above all submenus have a help-button now.
- The info button gives some extra information about the program and the loaded database
- In the COMMAND menu there is a 'statistics' option which is useful when dealing with databases with a lot of numbers
- Time consuming actions are accompanied by a "WAIT" info so you will know that the program is still working
- Scrolling does not stop at the end of the 'visible' records any more. The next lot will be shown so that you can see ALL your records now
- Accordingly there is also a 'jump to first/last record' feature
- Single view of a record: very much wanted by a lot of people!

and still more small things which are described in a ReadMe file you will get when updating. Really great, isn't it! Thank you BOB.

So all you people out there who are proud owners of **Suqcess**, get your update! Jochen Merz always has the newest versions. And all people who don't own it, don't hesitate, it's worth it's price! (By the way, the money will not be for me any longer or for Bob, the full amount is for Jochen as a reward for his unbelievable fidelity to the QL and his customers. Thank you Jochen!)

And thank YOU, Wolfgang! Usual update rules apply: Send your master disks together with International Reply Coupons to J-M-S. Jochen

CAMBRIDGE Z88

OFFICE/FAX 01494-871319 (EEC) *W.N. Richardson & Co.*

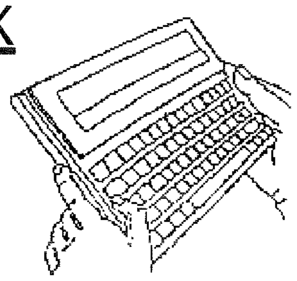
MOBILE: 07808 576118
E-MAIL: WNR@COMPUSERVE.COM

6 Ravensmead
Chalfont-St-Peter
Buckinghamshire SL9 0NB

THE IDEAL PORTABLE COMPANION FOR THE QL, PC AND MAC.

THE CAMBRIDGE Z88 A4 NOTEBOOK

WITH BUILT-IN WORD PROCESSOR, SPREADSHEET, DATABASE, BASIC, CALCULATOR, CLOCK, ALARM, CALENDAR & VT52 TERMINAL.
USES 4XAA ALKALINE CELLS (ca. 20 HOURS) ✱



WE DO REPAIRS, PART EXCHANGES, AND BUY Z88's & PARTS

NEW! CAMBRIDGE Z88 WITH OZ4+512k Internal Ram £150

ALSO AVAILABLE, 512k RAMPACK £40

DESCRIPTION	UK	USA
CAMBRIDGE Z88 ISSUE4-128K	**£120**	**$180**
CAMBRIDGE Z88 COMPUTER	**£99**	**$130**
☆**RECONDITIONED Z88**	**£60**	**$90**
32k RAMPACK	£18	$25
128k RAMPACK	£38	$50
32k EPROM PACK	£18	$25
128k EPROM PACK	£24	$33
256k EPROM PACK	£55	$75
EPROM ERASER	£32	$43
PARALLEL PRINTER LEAD	£39	$50
SERIAL PRINTER LEAD	£12	$16
✱ **MAINS ADAPTOR (230vac; 6v,500ma)**	**£10**	
TOPPER (PROTECTIVE COVER)	£12	$16
CARRYING CASE (PLASTIC)	£10	$12

1/2 PRICE STARTER PACK 1 - Z88 ☆ PLUS TRANSFER KIT + 32K RAM + 32K EPROM £72 ONLY!!

FILE TRANSFER TO OTHER COMPUTERS

	UK	USA
SPECIAL Z88-QL SERIAL LEAD	£10	$15
COPY DISKS OF QUANTA PROGS IMP/EXP & ARCHIVE EXPORT	£2	$3
PCLINK KIT (For PCs)	£25	$35
Z88 TO MAC KIT	£25	$35
Z88 TO BBC KIT	£25	$35

} ALL WITH INSTRUCTION BOOKS, CABLES AND EPROMS AS REQUIRED!

POSTAGE UK UP TO £5. EEC £15. USA £20. OTHER COUNTRIES £30.

ALL THE STOCK IS WARRANTIED FOR 90 DAYS. IN THE EVENT OF REPLACEMENT BEING ARGREED, BUT THE ITEM BEING OUT OF STOCK AT THE TIME, A REFUND WILL BE MADE PROVIDED THE ITEM IS RECEIVED IN GOOD CONDITION.

QL & PC COMPUTER USERS WILL FIND THE CAMBRIDGE Z88 ESPECIALLY USEFUL FOR WORK AWAY FROM THE DESKTOP, WITH TRANSFER PROGS DATA CAN BE SAFELY EXCHANGED WITH THEIR DESKTOP SYSTEM.

W.N.RICHARDSON & CO CONTINUES TO PROVIDE FULL SPARES AND SERVICES FOR SINCLAIR COMPUTERS, QL & THE CAMBRIDGE Z88

Programming Example for Xmenu with CPTR

J. M. Sadler

In Vol 7 Issue 4 of QL Today George Gwilt described the "orthodox" way of writing PE programs with TurboPTR or CPTR. This article is to show you how easy it is with CPTR, using as an example test4 from Part 4 of Jerome Grimbert's articles on XMENU in Vol 7 Issue 4 of QL Today.

If you have not installed CPTR, the first step is to unzip the program files for CPTR. The programs setz, spr and seewinf should be put in a suitable directory or floppy. The file libcptr_a should be put in the C68 LIB directory or floppy. The file wdef_h should be put in the C68 INCLUDE directory or floppy.

So that setz works properly set PROGD$ with prog_use "Location of CPTR" and DATAD$ with data_use ram1_

We are now ready to go through the process of producing the program "test4".

I have taken this step by step so that even a new QL user can easily achieve the final result.

A. Producing the Window by Using setz

The program setz follows the same course as setf whose operations are described in the Appendix to George Gwilt's article on PE windows in Vol 7 Issue 4.

```
Start program                                 ( Enter "ex setz" )
Program name is "test4"                       ( Enter "test4" )

Alter Text                                    ( Press n )
                                              ( Enter "PE in C test 4" )
                                              ( Press n )
                                              ( Enter "Quit" )
                                              ( Press Esc 4 times )

Number of main windows = 1                    ( Press enter )
Number of loose items = 4                     ( Press 4 enter )
Number of information windows = 1             ( Press 1 enter )
Number of loose items  = 1                    ( Press 1 enter )
Number of Application Windows = 1             ( Press 1 enter )
Number of menu items = 0                      ( Press enter )

For Main Window
The shadow size = 0                           ( Press enter )
The border size = 1                           ( Press 1 enter )
The border colour = black                     ( Press up arrow once & enter )
The paper colour = white                      ( Press up arrow four times & enter )
Sprite is default                             ( Press Esc )
Presentation of loose items is default        ( Press y )
Type of Loose item 1 is text                  ( Press enter )
Text is "Quit"                                ( Press down arrow & enter )
Key is Esc                                    ( Press Esc )
Type of loose item 2 is sprite                ( Press down arrow & enter )
Sprite is mode 8 (changed later)              ( Press up arrow twice & enter )
Key is Ctrl F1                                ( Press Ctrl F1 )
Type of loose item 3 is sprite                ( Press down arrow & enter )
Sprite is move                                ( Press up arrow 5 times & enter )
Key is Ctrl F4                                ( Press Ctrl F4 )
Type of loose item 4 is sprite                ( Press down arrow & enter )
Sprite is resize                              ( Press up arrow 6 times & enter )
Key is Ctrl F3                                ( Press Ctrl F3 )
Information window border is 0                ( Press enter )
Information paper is light green              ( Press down arrow & enter )
Object is text                                ( Press enter )
Object is "PE in C test 4"                    ( Press enter )
Ink is white                                  ( Press up arrow 4 times & enter )
X_csize is 0                                  ( Press enter )
Y_csize is 0                                  ( Press enter )
Application window border is 4                ( Press 4 & enter )
Application window border is grey             ( Press enter )
Application window paper is green              ( Press up arrow twice & enter )
Application window sprite is default           ( Press enter )
Application window select key is tab           ( Press tab )

Now for sizes of windows and position
Main Window is 180 wide by 230 deep            ( Alter until display shows correct size )
Variable sized window                          ( Press y )
Minimum size is 120 wide by 130 deep           ( Alter until display shows correct size )
Origin is 20 by 8                              ( Alter until position is correct & enter )
Loose item 1 is correct size at 90, 3          ( Alter until position is correct & enter )
Loose item 2 is 24 by 14 at 4, 16              ( Alter until correct & press enter )
Loose item 3 is 16 by 14 at 32, 16             ( Alter until correct & press enter )
Loose item 4 is 16 by 14 at 60, 16             ( Alter until correct & press enter )
Information window is at 2, 3                  ( Alter until position is correct & enter )
Object is in correct position                  ( Press enter )
Application window is 90, 80 at 14, 40         ( Alter until correct & press enter )

Now we position moveable items
Loose item 1 moves horizontally                ( Press n, n, y, n )
Loose items 2, 3 & 4 are fixed                 ( Press n, n, n, n, thrice )
Information window horizontal size             ( Press y, n, n )
Application Window is fixed                    ( Press n, n )
The information object is fixed                ( Press n )


Now the main window will be shown              ( Press enter ) . .
. . and the "sleep" window is shown            ( Press enter to complete the program )
```

You will now have four files in ram1_ test4_wda, test4_z, block_blo_bin & wh_pat_bin
test4_wda is the window definition file for using with TurboPTR. It can be displayed again with:
`ex seewinf;'test4'`

test4_z is the file to be edited and processed to create the source. The other files were used by setz and can be either left or deleted since they are no longer needed.

B. Editing the _z File

Open test4_z in your favourite editor.

Not all the parameters are needed in the function declarations alit0_0 alit0_1, alit0_2, alit0_3 and alit1_0 so change them to

```
alit0_0(void);
alit0_1(WM_wwork_t *,WM_litm_t *, WM_wstat_t *);
alit0_2(WM_wwork_t *,WM_litm_t *, WM_wstat_t *);
alit0_3(WM_wwork_t *,WM_litm_t *, WM_wstat_t *);
alit1_0(void);
```

We want to make the application window variable in size but setz did not allow this. We also want to use a hand sprite in the application window.

So in the lines following "static struct WM_dappw appw0 =":

In "static struct WM_dappw appw0 =" change

```
"  90,          /* xsize " to     "  90 + 16384,             /* xsize " (line 45)
"  80,          /* ysize " to     "  80 + 16384,             /* ysize " (line 46)
"  0,           /* pspr *" to     "  %%‹‹&wm_sprite_hand›,   /* pspr *" (line 53)
```

In "W_LITM (4, litm0)" change

```
"  0,           /* pobj *"  to    "  %%‹‹&wm_sprite_sleep›,/* pobj * " (line 110)
```

so we have a sleep sprite instead of something odd or a crash.

We have now finished tidying up the window definition and can start writing the program.

C. The Program

Scroll down to nearly the end of the file to just pass "End of Declarations" and enter the following before int main()

```
void (*_consetup)() = NULL;
char *_endmsg = NULL;
char _prog_name[] = "PE in C tutorial 4";
void (*_cmd_params) () = NULL;         /* No arguments are passed         */
long (*_cmdchannels) () = NULL;        /* Redirection is not used         */
long (*_stackchannels) () = NULL;      /* No parameters are passed        */

WM_wwork_t *wwa0;
chanid_t chid0;
```

We are instructing the compiler not to include facilities for parameters or redirection and setting the program name. CPTR uses the program name to display in the button when it is put to sleep.

*wwa0 is the global variable for the working definition.
chid0 is the global variable for standard out so the program to write to it.

The remainder, which immediately follows

```
int main()
 {
```

completes the program:

```
  int err;

  /* make  mode 8 mode 4 */
  short mode, type;
  mode = -1;
  type = -1;
  mt_dmode(&mode, &type);
  if (mode == 8) { mode = 4; mt_dmode(&mode, &type); }

  if(getsze(&wd0, prtab, &ws[0], wd0_sizes, &num)) exit (ERR_NC);
  if(!(chid0 = fgetchid(stdout))) exit (ERR_NI);
  if(!wm_findv(chid0)) exit (ERR_NF);
  if(!(wwa0 = malloc(wd0_sizes[0]))) exit (ERR_OM);
  if(wm_setup(chid0,0,0,&wd0,&ws[0],&wwa0,0)) exit (ERR_NI);
```

getsze creates a table to hold the pointers which the pointer environment uses to access various items and functions.
fgetchid and wm_findv connect chid0 to standard out and place it in the window. malloc creates space for the window definition, and wm_setup creates it.
Now add the lines

```
  if(wm_prpos (wwa0,DEFXY)) exit(ERR_NI);
  if(wm_wdraw (wwa0)) exit(ERR_BO);
```

wm_prpros positions the window and wm_wdraw draws it.
Then add the lines

```
  while(!(err = wm_rptr(wwa0))) ;
  free(wwa0);
  exit (err);
```

"while(!(err = wm_rptr(wwa0))) ;" reads the pointer until there is an error.
free(wwa0) frees the space for the window definition and "exit (err);" exits and displays the error if the program is started with ew.

RWAP SOFTWARE

PWord English Dictionary v1.0 £15

The ultimate UK english dictionary with over HALF A MILLION WORDS, compiled by Paul Merdinian who compiled the Mega dictionary for DP's Speelchecker.
Due to the sheer size, a Super Gold Card is the minimum requirement.
Two versions are available: QTYP dictionary only on HD disk - £10 CD containing ASCII list, QTYP version, Solvit Plus (from Just Words) and the dictionary in the solvit format, cost £15.

QL Cash Trader v3.7 £5

A well established accounts package for the small to medium sized business, including automatic generation of profit & loss account, balance sheet, VAT returns, reports and analysis for audit trails and management decisions. Previously sold for over £100.*

QL Payroll v3.5 £5

Manage a payroll for a small to medium sized business. Handles up to 99 employees easily, producing P45s and P60s as well as the payslips on a monthly or weekly basis. Calculates tax and national insurance and is easy to update to take account of the current tax year rules.

Sidewriter v1.08	**£10**
Image D v1.03	**£10**
Q-Help v1.06	**£10**
Q-Index v1.05	**£5**

Four excellent programs to assist the QL user.
Sidewriter: Produce landscape printouts on Epson printers.
Image D: Produce 3D pictures of objects.
Q-Help: on-screen help for SuperBASIC commands.
Q-Index: look up keywords related to topics.
See earlier adverts for more detials.

ProForma ESC/P2 Drivers v1.04 £8

New improved colour and monochrome printer drivers, providing up to 720dpi for all programs written for use with ProWesS, such as LineDesign and Paragraph. Works on all Epson inkjet printers which support binary mode compression (740, 850 and 900 models at least). 1440 dpi to follow.

QL Genealogist v3.26	**£20**
Genealogy For Windows	**£50**

Store your family tree for posterity. Add individuals with details of their parents and children, watch all of those links build up into a formal family tree layout. Text files and pictures may also be linked to individuals as well as notes and events, making this the perfect way to preserve the history of your family. QL version now supports FileInfo II and QMenu as well as allowing you to link both male and female trees. Sample tree of the Royal family since 1066 included.
PC version is event driven - enter the details as they appear in documents and it generates the tree from these. QL data and GEDCOM can be transferred to the PC version. Upgrade to latest PC version (v5.21) for £8
Both programs easy to use and complete with a step by step tutorial.
** QL USERS upgrade to PC version for £25 ONLY **

Q-Word *COMING SOON* £tba

The ultimate word game for the QL - you are given a grid of letters and need to link letters together to form as many words as possible. Points are based on both the number of words and their length, as well as letters used. As you use a letter, it is removed from the grid, with the object to clear the grid.
Using high colour graphics on all systems which support more than 8 colours (including Aurora), background music and much more, this will keep you entertained for a long time.

SBASIC/SuperBASIC Reference Manual £40
Updates £6 each, £10 for 2 (Current Version - Rel 4)

Have you ever tried to write a program, but been lost as to the means of performing a certain action? This Reference Manual provides you with a full description and examples of how to use all of the keywords found on each of the different QLs, plus SMSQ/e, Toolkit II and many different public domain toolkits. Details of any possible problems are provided, together with descriptions of how to use the device drivers and how to ensure that your programs are compatible across the range of QL platforms.
This book is ideal for all QL users and is kept up to date with regular updates. ** Currently Out of Print **

QL Cosmos v2.04 £5

Ever wondered what the stars in the sky looked like 100 years ago? Or, maybe you want to learn the constellations and names of what you see in the sky. This is the program for you - generates pictures of the stars and planets for any given place or time and provides details on these objects. Includes Halley's Comet, the Moon and the Solar System planets.

Q-Route v2.00	**£25**
Upgrade from v1.xx	**£5**

The latest version of this popular route finding program. Find the quickest route or the shortest route between any two places, using roads. A wide range of maps is available for this program (see elsewhere in this advert). The program is easy and quick to use. You can even add your own places and roads to the maps to include local detail.

Flashback SE v2.03 (Upgrade only) £5

The ultimate database program - extremely fast and flexible, easy to use, updated to cope with the latest versions of the QL operating system and still maintained. A report module is included to allow you to format output in any way, including mail-merge. Unfortunately only available as an upgrade from the original version (original still available from Sector Software).

Return To Eden v3.08	**£10**
Nemesis MKII v2.03	**£8**
The Prawn v2.01	**£8**
Horrorday v3.1	**£8**
West v2.00	**£5**
The Lost Kingdom of Zkul v2.01	**£5**

A wealth of QL adventures - mainly text only.
Save the Galaxy from the ambitions of the evil dictator Nemesis.
Battle against werewolves and dracula look-alikes on a Hammer Horror set in the comical Horrorday.
Take the part of a prawn with a hangover, lost in a strange land in the hilarious Prawn.
Solve a bank-robbery by fighting the bad guys and collecting the loot in real-time old West.
Battle countless dwarves in the atmospheric Lost Kingdom of Zkul.
Return to Eden is a massive adventure over 3 disks with colourful graphics - control 3 characters in their quest to find the missing Prince.
All six adventures are available together for only £25.

A range of games to keep both the young and the young at heart amused. Some are old favourites, like Golf and a pub quiz program (500+ questions). Others are fast, colourful arcade games. Flight simulator also now available. Plenty of variation and skill required - what more can you ask for?
All 6 programs only £28.

Open Golf v5.20	**£8**
QuizMaster II v2.07	**£5**
Stone Raider II v2.00	**£5**
Hoverzone v1.2	**£5**
Deathstrike v1.5	**£5**
Flightdeck v1.05	**£10**

The latest maps for Q-Route. Maps of various areas of Britain have been created by cutting them out of Big Britain Map - they will use less memory and can contain more detail. Areas covered: Scotland, NE England, NW England, S&W Yorkshire, Wales & Derbyshire, London area and South England.
Latest version of Q-Route is recommended.

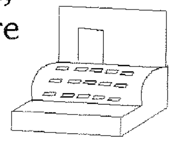
Britain.map v1.11	**£2**
BIG Britain Map (needs 2MB) v2.03	**£5**
Various Britain Area Maps (ask for details)	**£2 ea.**
Ireland Map v1.00	**£5**
Belgium Map v1.01	**£2**
Catalonia Map v1.03	**£2**

D-Day MKII v3.04	**£10**
Grey Wolf v1.8	**£8**
War In The East MKII v1.24 (Upgrade Only)	**£5**

For the gaming enthusiast - D-Day is a classic table top wargame for one or two players - you control either the Allies or the Axis forces during WWII. With the ability to define your own army set ups and a choice of 4 different scenarios, this should keep you entertained for a while.
Grey Wolf is a graphical simulation of a submarine - can you sink the enemy shipping whilst avoiding their planes and destroyers??

* Also known as Trading Accounts

RWAP Software, 35 Chantry Croft,
Kinsley, Pontefract, West Yorkshire
WF9 5JH
TEL: 01977 610509
http://hometown.aol.co.uk/RWAPSoftware



Cheques in £sterling
payable to 'R.Mellor'

After the closing bracket "}" we need a function for alit0_0 the "Quit" button or loose item.

```
/************************* Quit ****************************/

int alit0_0(void)
 {
  exit (EXIT_SUCCESS);
 }
```

Perhaps now is the time to explain an unique feature of C programming for the pointer environment. When a loose item is activated there are useful items in the registers. So instead of just jumping to the correct function the wrapper makes these items available so the function can use them. Hence if these items are declared in the function properly they will automatically be used by the function and the programmer does not have to worry that he has used to correct variables. So "static struct WM_action afun0_1 ={JSR,wm_actli,alit0_1};"
creates an area in the program which has the code for jump to sub routine actli and then jump to routine alit0_1. actli is a function which get these items for the function alit0_1. Then the function alit0_1 and uses these items if they are declared in the correct order and used in the function.

So now we need a routine for the sleep loose item.

```
/************************** Sleep ***************************/

int alit0_1(WM_wwork_t *wwk, WM_litm_t *li, WM_wstat_t *wst)
 {
  wst->evnt |= PT_ZZZZ;
  do_sleep3(wwk->chid, wwk,0, _prog_name, NULL, wd0_sizes);
  return slitem(wwk, li->item,0);
 }
```

wst->evnt |= PT_ZZZZ activates the loose item. do_sleep3 puts the program to sleep and slitem resets the loose item to available again.

Then a routine to move the window.

```
/************************** Move ****************************/

int alit0_2(WM_wwork_t *wwk, WM_litm_t *li, WM_wstat_t *wst)
 {
  short dx,dy;
  int err;
  wst->evnt |= PT_WMOVE;
  if (err = wm_chwin(wwk, &dx, &dy)) return err;
  return slitem(wwk, li->item, 0);
 }
```

wst->evnt |= PT_WMOVE activates the loose item and wm_chwin changes the window to its new location.

Next we need a routine to resize the window

```
/************************* Resize ***************************/

int alit0_3(WM_wwork_t *wwk,WM_litm_t *li, WM_wstat_t *wst)
 {
  short ddx, ddy, dx, dy;
  int err;

  wst->evnt |= PT_WSIZE;
  ddx = wwk->xsize+wwk->xorg + wd0.wdefa.xorg;
  ddy = wwk->ysize+wwk->yorg + wd0.wdefa.yorg;
  dx = 4*((dx + 3)/4);
  dy = 2*((dy + 1)/2);
 if (err = wm_chwin(wwk, &dx, &dy) < 0) return err;
 dx = EVEN(MIN(wd0.wdefa.xsize, MAX(wd0.wdefb[0].xsize &
                    0x3FFF, wwk->xsize-dx)));
 dy = MIN(wd0.wdefa.ysize, MAX(wd0.wdefb[0].ysize &
                    0x3FFF, wwk->ysize-dy));
 if (err=wm_unset(wwk)) return err;
 if (err=wm_setup(chid0, dx, dy, &wd0, &ws[0], &wwk,0)) return err;
 if (err=wm_prpos(wwk, ddx- dx, ddy- dy)) return err;
 if (err=wm_wdraw(wwk)) return err;
 return slitem(wwk, li->item,0);
}
```

After activating the loose item with wst I= PT_WSIZE, the absolute pointer position is placed in ddx & ddy.
Then new size is placed in dx & dy and rounded up to multiples of 4 & 2 respectively.
The new size is restricted to the maximum size and minimum size of the window.
wm_unset removes the window, wm_setup creates again but has to find channel from chid0, pr_pos positions it and wm_draw draws its, but not its contents.

Finally add the following

```
/******** Dummy action routine for the button window ********/

int alit1_0(void)
 {
  exit (EXIT_FAILURE);
 }
```

This finishes editing the z file.

D. Compilation

Now convert ram1_test4_z to ram1_test4_c file with

```
 ex spr,#1;'test4'
```

If all has gone well you will have a new file test4_c. If not the error will be shown on #1.

Now if the location of your C68 directory is not the same as your CPTR directory change it with

```
 prog_use " Directory of C68 "
```

Compile with

```
 ex cc;'test4_c -l cptr -o test4'
```

and check it with

```
 ex ram1_test4
```

E. Conclusion

George's CPTR method is so much easier for designing windows and also a lot faster than any other I have seen. Furthermore you have the advantage that it uses the pointer routines and so there is no problem of changes in the pointer environment making it out of date or in the resizing of windows. The code seems to be more compact. There are further explanations in George's test files and examples. There are far more facilities inside CPTR such as a method for designing sprites, the ability to display help comments, drag & drop, and scroll bars for application windows. Finally the code is actively maintained and extended when required by George so the more you use it and ask for extensions the more likely it will be extended to suit your needs.

Programming with QPTR - Part 5 - The level III pointers

Wolfgang Lenerz

We finished the level III pointers for information subwindows last time. Now it's time to explain those for application subwindows:

B - Application subwindows

There are two kinds of application subwindows. First, there are "**menu application subwindows**". These contain elements which behave like loose menu items: they can be selected, clicked and actioned. A typical example would be the QPAC2 "Files" menu - the names of the files which are displayed are part of an application subwindow: they can be selected, and, if you DO them, they produce an action. The "menu items" of the menu application subwindows are displayed in a grid. This makes a nice contrast with the loose items.

The second type of application subwindow is the "**simple**" application subwindow. This does not contain a menu, in fact it is empty. Since it doesn't contain anything, it is easier to define than a menu application subwindow.

As we have seen for the LEVEL I definitions, there is a list of application subwindows, composed of pointers (addresses) to the subwindow definitions. This thus must mean that application subwindows also have definitions... and, indeed, there is one definition per application subwindow. This definition is built with the ***MK_APPW*** (***M**a**K**e **APP**lication sub-**W**indow definition) function:

```
appsubwin = MK_APPW(awdef%, aattr%, aptr,
akey$, x_ctrldef, y_ctrldef, xoff%, yoff%,
x_spac, y_spac, xindex, yindex, linelist)
```

Phew!

Let's start by the easiest bit: The first four parameters. The first two of these (i.e. awdef% and aattr%) are identical to the first two parameters of the ***MK_IWL*** function which was described in the last instalment of this series. They determine the "physical" definition of the window (adef%) and the window attributes (aattr%) as follows:

-› * **awdef%** is an array containing the physical description of the application subwindow. It has a dimension DIM (3). The array contents are:

  - window x size (element 0)
  - window y size (1)
  - window x origin (2)
  - window y origin (3)

  The origins are the top left corner of the window with respect to the top left of the primary (or secondary) window containing the application subwindow.

-› * ***aattr%*** is an array with the attributes of the subwindows. It is again an array DIM (3). The array contents are:

  - Shadow "depth" - this is actually ignored for application subwindows and should be left at 0.
  - border size
  - border colour
  - paper colour

  of the application subwindow, in that order.

-› * ***aptr*** is the address of a pointer sprite for this application subwindow. Thus, each application subwindow may have a pointer sprite that is different from the main window pointer sprite!

-› * ***akey$*** is the "selection key" of the application subwindow - this is used to bring the pointer directly into the application subwindow. Moreover, if the application subwindow is a menu application subwindow with a scroll bar hitting this key will bring the pointer:

  - first to the centre of the application sub-window, if the pointer was not already in the application sub-window.
  - then , if you hit it again, onto the scroll bar (if any!)
  - then back to the centre of the application subwindow
  - and again onto the scroll bar - and so on...

  Just like for loose menu items, this selection key must be passed to the MK_APPW function in upper case. Generally, the TAB key (chr$(9)) is used, if there is only one application subwindow.

It is possible to define application subwindows with these first four parameters only. In this case, we have a simple application subwindow, and the call to RD_PTR (see below) will come back each time the pointer has moved or a key was hit (provided, of course, the pointer was in the application subwindow!).

If, however, you wish to define a menu application subwindow, you must fill in more parameters which will be explained below.

IV - LEVEL IV: Defining rows and columns

Before starting on this, let's see what a menu application subwindow consists of. This is one of the most complex aspects of QPTR programming - again, it is not difficult, there are just many parameters to learn (and remember)... However, if there are many parameters, this also means that you will have a large freedom to set up these windows (else the parameters wouldn't be of any use).

A - The components of a menu application subwindow

As we have seen above, the first parameters of an application subwindow are normal: size and origin of the subwindow, colour and size of its border, pointer, "paper" colour and "selkey". These parameters shouldn't be complicated.

Apart from that, an application subwindow is nearly entirely composed of "objects", i.e. the items of the menu. As mentioned, these are similar to loose meny items, but are arranged in a grid of rows and columns.

If need be, one my also add the scroll/pan bar and scroll/pan arrows. You can clearly see this in the "Files" menu of QPAC 2 , where all of these elements are visible. The "objects" are, of course, the filenames.

Just as a reminder: when the window can be scrolled up and down, then this is a "scroll". If the same is possibel for left to right, then this is a "pan".

1) The objects

As mentioned, the objects are items, quite similar to loose items. Here again, you must make a list of these objects and specify the type of each object (text, sprite etc...). In most cases it will be text, but not necessarily so, as Jérôme Grimbert shows in these hallowed pages of the august magazine (see his series on XMenu).
For each object, you also specify a possible selection key, the content type (i.e. the text), the content itself and the position in the grid. As you can see, quite a long number of parameters. This list is made up of Level V parameters which will be detailed later, and is built with the MK_AOL function.

Let's suppose for now that the list has already been built and that "objlist" is the result of the MK_AOL function.

Since the objects of an application subwindow behave similarly to loose menu items, the current item is also surrounded by a border, like the current item in a loose menu. The objects can be selected, thus changing their status, and if an object is "done", it may produce an action.

Here again, like for menu items, you will have to determine the **attributes** of these objects: the colours for the different statusses, and the colour and size of the current menu item border. These are common for all items of an application subwindow. Of course, different application subwindows may have different colours (I'm not sure whether that would be a good design practice, though).

This is where the similarity with loose menu items ends, as here we do not have "loose" items, but "bound" items - they are bound to each other and part of a grid of rows and columns.

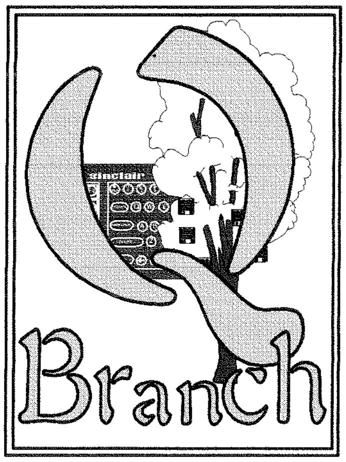
2) Columns and rows

Since the items are part of a (hopefully regular) grid, we must specify how these objects are to appear in the grid. Whilst this is necessarily in rows and columns, you can specify how many rows and columns there are to be. The columns for each row need not be identical.

In most cases, the most important element is the row. You must determine which object(s) can be found in which row. Thus you must establish a **row list** which clearly states what row contains which objects, e.g. show that it contains objects a to b. The next row then contains objects c to d etc... If there is only one object per row (e.g. The QPAC 2 "Files" menu with "Statistics" switched on) this is not really complicated: you just indicate that object 1 is in row 1, object 2 in row 2 , 3 in 3 and so on.

If there are two objects per row, you will indicate that objects 1 and 2 are in row 1, objects 3 and 4 in row 2, 5 and 6 in row 3... - you get the picture.

Q Branch



Text 87

£ 79.00

Typset 94	£ 29.00
Fountext 94	£ 39.00
2488 drivers	£ 29.00
Epson ESC/P2 drivers	£ 26.00

Text 87 is the only QDOS / SMSQ wordprocessor capable of handliing the full screen on the Aurora / QXL / QPC systems. New drivers are currently being written.

PROGRAMMING

QD 98	£ 45.00
QD + QBasic	£ 59.00
QD + Qliberator + QBasic	£ 100.00
Qliberator	£ 50.00
Master Spy v 3.3	£ 30.00
QPTR	£ 30.00
Easyptr pt 1 & 2 (together)	£ 30.00
Easyptr pt 3 (C library)	£ 14.00
QMake	£ 15.00
QMon / JMon	£ 22.00
Basic Linker	£ 19.00
DISA 3	£ 31.00
QMenu	£ 14.00

Spring Colours

Changes are afoot in the SMSQ camp. A new version of QPC 2 with full support for the colour palette in the Window Manager will be released soon. There will also be a release of new versions of QMenu, QD and QSpread. At the time of going to press these are still at the Beta testing stage. Contact us for details.

New Keyboard Membranes !!

Available soon - £17.50 + p&p

The BlaQbox - fast PCs in black boxes - call for details.

'Just Words' by Geoff Wicks

THESAURUS, STYLE CHECK

£ 10.00 ea / ANY 2 PROGRAMS £ 18.00 / ALL 3 PROGRAMS £ 25.00
(Includes Pointer and non-pointer driven versions)
(P.E. versions need Hot_rext, WMAN and PTR_GEN or SMSQ/E to run)
Upgrades from previous versions £ 2.50 + S.A.E. New Manuals £ 1.50

QL2PC

Convert text files from QL to PC formats and much more !
Only £ 10.00 Now with HTML support !

QL Rhymes £ 10.00
Rhyming Dictionary

Spelling Crib : PD program £ 1.50 +SAE
or Free if you buy all three programs

UTILITIES

FiFi 2	£ 18.00
QSup	£ 28.00
QSpread v 3	£ 48.00
Cueshell 2	£ 15.00
Qload / Qref	£ 15.00
Disk Mate 5	£ 16.50
QPAC 1	£ 20.00
QPAC 2	£ 40.00
QTYP 2	£ 30.00
QLQ	£ 28.00

We are currently out of stock of the SuperBasic Reference Manual Place your order now to get one as soon as it is reprinted

The SBASIC / SuperBASIC Reference Manual

The complete definitive guide to BASIC programming in QDOS / SMSQ including three disks of PD toolkits, example procedures and an electronic index.
compiled by Rich Mellor, Franz Hermann and Peter Jaeger

Over 500 pages !
£ 40.00
+ postage

News on the SuperBasic Reference Manual Coming soon !

Tel +44 (0) 1273-386030 fax +44 (0) 1273-430501
Mobile +44 (0) 7836-745501
email qbranch@qbranch.demon.co.uk web : http://www.qbranch.demon.co.uk

Q Branch

Q Branch

Feeling out on a limb ?
Reach out for Q Branch.
Suppliers of Quality QDOS/SMSQ products
Hardware and Software.

20 LOCKS HILL, PORTSLADE,
E. SUSSEX. BN41 2LB. UK.

New Fax Number

ProWesS

ProWesS (now free !)	£ 1.60
DATAdesign	£ 20.00
Fontutils	£ 28.00
File Search	£ 11.00
PFlist	£ 11.00
Dilwyn's Fontpack	£ Call
LINEdesign v 2.16	£ 22.00
PWfile	£ 17.50

Paragraph

The ProWesS word processor
¶ Demo version £ 1.50 + postage
Full Registered version £ 18.00
Version 2.03 available now !

Hardware

We have a small stock of second user items. Auroras / Qubides / Gold Cards / Qplanes / superHermes etc. call us to get details of the items available. These are going fast so call soon.

QXL II	**£ 100.00***
Recycled superHermes	£ 65.00 *
Recycled Gold Card	£ 45.00 *
Recycled Aurora	£ 70.00**
Qubide	£ 50.00*
Qplane	£ 5.00
Aurora cables	£ 3.00
Aurora rom adaptor	£ 3.00
'Arfa Braquet'	£ 8.00
'Son of Braquet'	£ 18.00
The 'Braquet'	£ 16.00
MC plate	£ 6.50

* when available.

Qubide upgrades to version 2.01 £ 8.00

SMSQ/E

Gold Card / Atari / QXL Version

Now Only £ 32.00

Various Atari versions : call for details

QPC 2 v3 full colour version!

Now Only £ 65.00

Upgrades from QPC 1 £ 46.00 (return master disk)
Upgrades from QPC 2 v1 £ 30.00
Upgrades from QPC 2 v2 £ 14.00
If Bought/upgraded after 15/10/01 £ 5.00

Special offer !
Get Cueshell for only £15.00 with any copy of SMSQ/E or QPC 2 upgrade !

Q Branch Programs

The Knight Safe 3	£ 35.00
upgrades from previous versions	£ 5.00
Q - Count Pointer driven home accounting	£ 25.00
Q - Route v1.08C Route finding programme	£ 25.00
The Fractal Collection	£ 35.00

Post and Packing is included with Software Prices

VISA

We can accept payment by Visa, Mastercard and Switch. We also accept cheques drawn on a UK Bank in Sterling.
You can also pay directly into our bank in Euros.
Barclays Bank Acc# 79016888 sort code :20-71-02 Name :QBranch)

MasterCard

(This row list is made with the MK_RWL function, commented below).

So, by now we will have indicated the content and parameters of each object, and in which row each object is going to go. Now you have to determine the size of each row, by determining the size of each column.

Each column as two sizes: the 'hitsize' and the 'spacing' between objects. There is one of each per column in the row.

The hitsize is the maximum size of an object in one column of the row - this actually defines the column size. It is this size that will change colour according to the status of the item. Again, look at the "Files" menu - if you click on a filename, it is not only the paper under this filename that changes colour, but the whole area that goes up to the second column (if any), and this, whatever the length of the filename may be. It is also that area which is outlined by the border when you bring the pointer over it, showing that this is the current item.

The "spacing" determines the number of pixels between the beginning of the hitsize of the first column and that of the hitsize of the next column, if any (and then the next columns, if any etc...). Clearly, the spacing must be at least as large as the hitsize, and ideally a bit larger (so that the border around the current item can be shown).

Let's presume that we have four rows with three columns each. And let's further suppose that the objects in the second column will be longer than those in the first column. I could then define the hitsizes and spacings as follows:

column one : hitsize 50, spacing 54
column two : hitsize 70, spacing 74
column three: hitsize 40, spacing 44.

The numbers correspond to the sizes in pixels: the first column will have a hitsize of 50 pixels and a total space (spacing) of 54 pixels. There will thus be at least 4 pixels between the object in that column and the object in the next column. You should make sure that the column is at least as long as the largest object that can go into it. If not, the object will be cut (if it is a text) or even not drawn at all (if it is a sprite).

You determine the hitsize and spacing for each column of each row. By doing this, you build up what is called the "**spacing list**". There is no need to have each column in each row to be the same size as that of the rows above and below. You don't even have to have the same number of columns in each row. Again, I consider it to be good programming practise to have a regular grid. It does make presenting the data easier.

It seems obvious that if you add up the spacings of each row in each column, you should get the size of the subwindow. It is possible, however to exceed that size, in which case the application subwindow becomes "pannable".

Likewise, if the combined height of all rows exceeds the height of the window, the window becomes scrollable.

3) Sections

An application subwindow can be cut up into several independently scrollable (or pannable) "sections". Each section can be scrolled independently, but they all show potentially the same data.

Sections are not necessary for application subwindows. If you take the QPAC2 Files menu for example, there are no sections. Let us suppose there were, though. If you have so many filenames that the window becomes scrollable, you could cut up the window into two sections. The window would be split up horizontally into two sections, there would be scroll bars for each section.
In principle, each section has its own scroll bars/scroll arrows (and pan bars/pan arrows of course). That way, you can see, at the same time, the start and the end of your data (in this case, the filenames.

It is important to realise that all sections may use the same rows and columns, and thus you can see all of the data in each section - you just have to scroll through it.

The user doesn't have to use the same different sections. In general, when the user brings the pointer to the scroll bar (NOT the scroll arrows) and "does" on the place where the two sections come together, the sections are joined and become one.

4) The control definition

The **control definition** tells the pointer Environment:

* how many sections there are
* how many rows there are in each section
* at what row each section starts
* where each sectionstarts in the window

OK, we have now seen the different elements that make up the application subwindow. So let's start defining them.

B - The parameter

First of all, you may have noticed above that two parameters to the **MK_APPW** function were left unexplained:

-› * **xoff%** This parameter just gives the number of pixels between the left border of the window and the first object on the left of the window. This applies the the first column of all rows of the applicaton subwindow. If left at 0,the first object will be right up against the left hand side of the application subwindow.

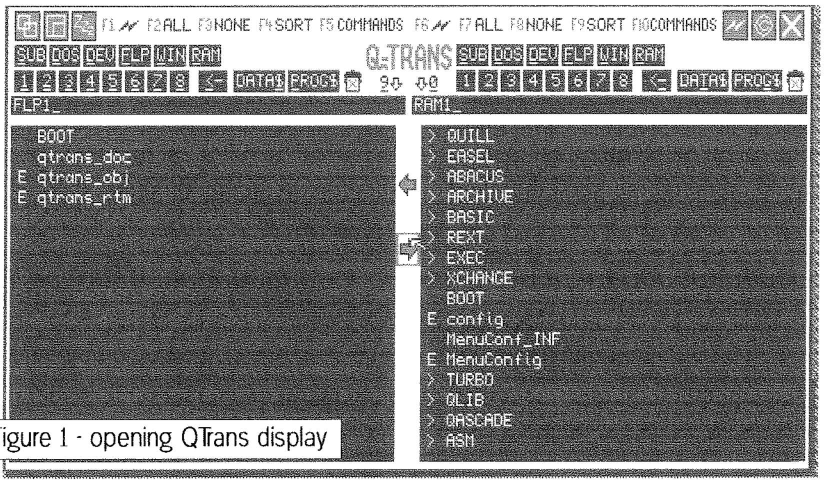
-› * **yoff%** is the distance, in pixels, between the uppermost visible row and the upper border of the application subwindow.

Ok, that's it for this time. Next time, we'll continue looking at level IV parameters.

QTrans Review

John Perry

QTrans is a quaintly named file copy and transfer utility. While QL file handling programs are ten a penny, this is one of the ones which does stand out from the others. This review is of version 1.03 which had just been released at the time of writing this review. In fact, it was the latest in a flurry of releases.
For a start it's pointer driven. It's claimed to be a precursor to a full blown GUI (Graphical User Interface) for QL systems. And it has quite a few novel features, like the dual file listing windows enabling you to see simultaneously the list of files on both the drives you are copying files from and to. In addition it has all sorts of commands and facilities for just about any file action from viewing and printing to searching and trashing.
Trashing? Well, one of the novel features of this program is the Trash Can. This is a facility which lets files be deleted, but done so in a way that allows you to undelete later.
It's a fairly rudimentary form of 'recycle bin' or similar facility found on other computers. It is not a true 'delete' action but rather files are put into a special folder on a hard drive rather than being deleted as such. In fact, there is a choice of Delete or Trash commands, meaning you can choose how files are deleted (provided you remember to use the correct command of course!)
The author suggests you give this folder a short and unusual name such as WIN1_*_ or a single letter name if you prefer something easier to remember. I opted for WIN1_*_ as it made it less easy for me to use a command from BASIC, for example, to accidentally delete something from this folder! In use, it worked well enough even if the Trash directory seemed to fill up at an alarming rate the way I go through files! In fact, for each file trashed, it seems to create two files, one with the original filename, and another much shorter file with a filename suffix of _T which contains details of where the file came from and the original file dates - yes, it even preserves file dates if that's important! The content of the Trash directory can be viewed just like any other directory on your hard drive and restoring files is as easy as copying files normally. Either navigate to the Trash Can folder or simply hit or do on the little icon of a bin, then select the Untrash command and it'll offer the choice of whether to restore the file to the original directory it came from or to the current path, which is what it calls the directory content shown in the other window.
I've realised I'm letting my enthusiasm get ahead of me here, so let's start with a screen dump from the program to show you the basics of what this program is all about.


F1 F2 ALL F3 NONE F4 SORT F5 COMMANDS F6 F7 ALL F8 NONE F9 SORT F10 COMMANDS
SUB DOS DEV FLP WIN RAM
Q-TRANS
SUB DOS DEV FLP WIN RAM
1 2 3 4 5 6 7 8 <- DATA$ PROG$
1 2 3 4 5 6 7 8 <- DATA$ PROG$

```
FLP1_
  BOOT
  qtrans_doc
E qtrans_obj
E qtrans_rtm
```

```
RAM1_
> QUILL
> EASEL
> ABACUS
> ARCHIVE
> BASIC
> REXT
> EXEC
> XCHANGE
  BOOT
E config
  MenuConf_INF
E MenuConfig
> TURBO
> QLIB
> QASCADE
> ASM
```

Figure 1 - opening QTrans display

The basic colour scheme is black and white with a little bit of red thrown in. At least this is a bit better than the early beta-test releases which were mostly bright green, which did not appeal to me at all. Six red icons across the top provide the usual program controls for pointer driven program of Move (move program around the screen), resize (anything from about half QL screen up to 1024x768 which was larger than my system would allow me to test), a ZZZ icon to make the program into a button, a redraw icon, a copyright notice screen and an X icon to exit from the program.
Between these are function key commands to let you read content of a directory again, select All or None of the filenames for copying etc, sorting the list of files and a Commands menu. If you are used to the QPAC2 files menu, you will find these are not the same keyboard shortcuts, indeed, you may find that such slight differences to what you are used to may make the program feel awkward at first.
Beneath these are a QPAC2-style list of device names, FLP, WIN, RAM, MDV etc. It only seems to list a maximum of 10 devices, and only those which actually exist on the system in question. I suppose if you had a TF Services ROMDisk it would add ROM to the list. DEV seems to be listed, I suppose Qubide users would get SUB as well (incidentally, while mentioning Qubide I should also say that the Trash Can is not the same Trash Can as that given with later versions of Qubide, this common name is probably a bit unfortunate and confusing).
Beneath these are QPAC2 style icons for drives 1 to 8. So if you want to display a list of files on FLP2_, you could use the FLP icon followed by the 2 icon, or even type in FLP2_ in the wide box beneath the numbers. There are also icons for the DATA_USE and PROG_USE settings (toolkit 2 default drives). Additionally, ‹- icon lets you go back down a directory, e.g. if you are viewing files in WIN1_QUILL_ and you wish to go back to WIN1_ just hit the ‹- icon and it seems to go back to the previous underline character in the path name. I've always struggled a bit with QL directories so this is not the easiest subject for me, but Q-Trans did seem to do what I expected it to, with one exception. If a directory didn't exist, it seemed to go back to the basic drive name rather than back to the next level down. The best example I can think of is if I asked it to list WIN1_ABACUS_FILES_ but that didn't exist, but WIN1_ABACUS_ did, it seemed to go back to WIN1_ instead. The author tried to explain that if a directory was not found, the program defaulted to root drive which is not unreasonable, just a bit hard to get your head around.
That apart, set the two drive and directory names you want to see the list of files for. Highlight a few filenames (if you hit a filename it turns red to highlight a selected file) and click on the red arrows between the two windows and it copies the files and updates the list of files after copying. It's that simple I keep asking myself why nobody's done this before for the QL!
The list of files can be quite long, I've tested it with a hard disk with just over a hundred files in a given directory. It seems to cope quite well, although it takes a while to read a very long list of files, even longer if you ask it to sort them. If there's more files in the list than fits the black window, the usual scroll bar arrows appear.
Filenames have symbols before them which again match the QPAC2 ones to indicate the type of file - the '›' symbol shows that the name is a directory, an 'E' shows it's a program which can be EXEC'ed, and there is another symbol for SROFF files whatever they are.
If you move the pointer over a directory name and either press the right mouse button or ENTER (Do is the QL term) the window changes to show the content of that directory (again, like QPAC2 files menu), making it quite easy to go from directory to directory in conjunction with the '‹-' icon. If you right click on any other filename, it brings up the command menu, rather like pressing F5 or F10 (F10 on a PC key-

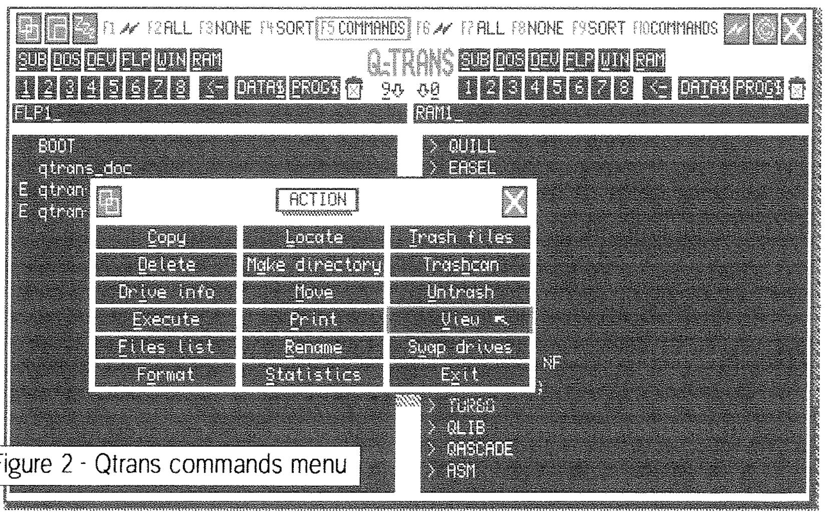
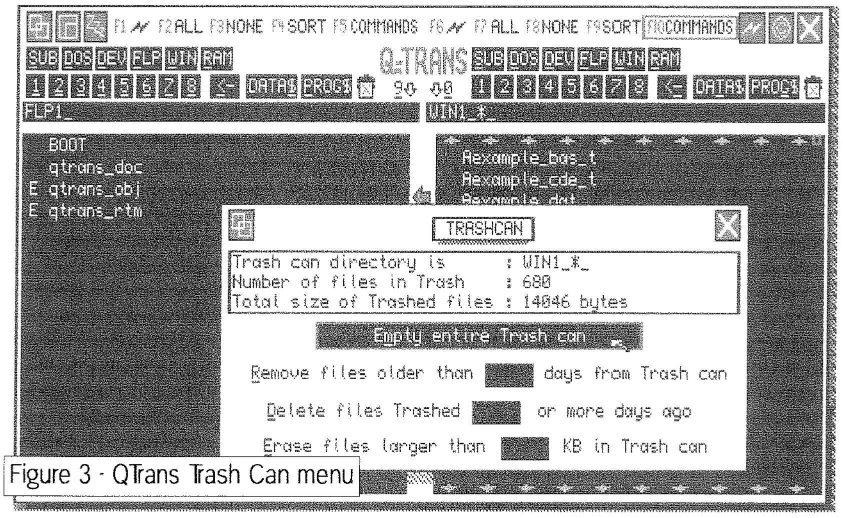


Figure 2 - Qtrans commands menu

board corresponds to shift f5 on a QL keyboard) for either window. Figure 2 shows this commands menu.

This menu contains the usual file copy, move, rename, delete, make directory, format and view commands along with a few less common ones. The Drive Info command shows how much space is free and used on a drive and whether the floppy disk is qdos or dos format on systems which support reading dos disks (i.e. smsq). I should mention that Qtrans runs on qdos and smsq/e systems.
The Execute command is very comprehensive, as long as you understand what all the options do. You can send command strings to programs, set the hotkey style options for "naughty" programs like Quill, even execute several programs in succession if more than one was selected. You can even execute a text file or a basic program if you have File Info 2 installed on your system - this lets you "associate" a program with certain types of files which have certain filename endinfs, e.g. you could set it up so that if asked ot execute a text file, what it actually does is to load your favourite editor and force that to load the text file, in much the same way that a Windows system might fire up Word to load a .DOC file for example. It's a pretty complex process, but worth mastering as it gives you a great deal of control over your system.
The Files List command simply produces a printed list of files, letting you specify where to print to and if a form feed is needed at the end to eject the last page from the printer. There is also a matching Print command which lets you print text files to a printer. Sadly, no facility to print screen dumps of graphics or other non-text files.
The Locate command is fairly complex. It searches all files in the specified directory (but not sub-directories sadly) for filenames or files containing given text. You can enter text you know is contained in a filename, or text contained in the body of a file or both and it will (fairly slowly) hunt for those files. Once found, you can view the file using either the built in text file viewer or if you have File Info 2 you can view the file with whatever program File Info 2 is set up to use to view that file. Again, complexity, but worth it if you are able to master it. Less obvious is that once it finds a file, a button called Locate has to be hit to find the next file containing that text. In addition to the boxes letting you enter text to be searched for, there is a box called 'Searching In' which mystified me at first as no matter how much I clicked on it, nothing seemed to happen. Soon I realised this wasn't an entry box at all, but shows the file the program is scanning during the search. While it works well enough I felt this should perhaps have been a different colour or layout to make it clearer that this wasn't a box expecting you to type in some information. The Locate command is not the fastest or most flexible ever, but does come in useful for simple searches.
QPAC2 users will know that it has a Move command which can be used to either rename a file or copy it to another drive and delete the original afterward. QTrans has separate Rename and Move commands. Rename has to be used if the file is simply being renamed on the same drive, but if you wish to change a filename to move it into a different directory you have to use Move.
Statistics displays information such as length of a file, dataspace of programs, file dates etc, the sort of information you'd get with a WSTAT command in BASIC for example. Annoyingly, it only displays details of one file at a time, though if you select several files it's very easy to step through them using the Next and Previous icons. QPAC2 has a Statistics command which shows the statistics alongside the filenames. Qtrans can't do this, as it has a fixed width layout with each window being just wide enough to show the longest possible QL file names, so I suppose it has to be done this way. It does take a bit of getting used to, though.
I've touched on the Trash Can facility already. In fact there are



```
Trash can directory is        : WIN1_*_
Number of files in Trash      : 680
Total size of Trashed files   : 14046 bytes

Remove files older than      days from Trash can
Delete files Trashed      or more days ago
Erase files larger than      KB in Trash can
```

Figure 3 - QTrans Trash Can menu

a number of options you can set for the Trash Can using the similarly named command. This brings up a menu (shown in Figure 3) which lists how many files are in the can and how much space they take (it's quite slow to read the space if there's a large number of files, I suppose it must read through all the files each time to check this).
From this menu, you have a choice of four ways of emptying or part empying the can. The crudest is simply to empty the whole lot in one go. This might be most useful if you have just done a full backup of your hard disk, for example, or if you have run out of space altogether. The other options let you remove files older than a given number of days, or files trashed (the term the author uses for files deleted and moved into the Trash can) more than a given number of days ago (the difference is that the former option seems to be based on what the original file date was, i.e. how old the original file is rather than how long it's been trashed), and files larger than a given size can be erased as well. Once removed from the can, there is no going back, trashing an already trashed file is final.
The Swap Drives command simply lets you swap over the content of both windows, e.g. if the left one is showing FLP2_ and the right one FLP1_, you can reverse these if that makes more sense without having to select each one in the usual way.
The program is configurable with the usual Config program. You can set options like which pair of drives are listed as the program starts. This can also be over-riden by passing the drive names in an EX command separated by a space as the program is started:

```
EX FLP1_QTRANS_OBJ;'FLP1_
RAM1_'
```

starts it displaying the content of FLP1_ and RAM1_ rather than what is configured. I'm not sure how this would be useful, unless another program needed to call up QTrans for some reason, it might be something to do with the Desktop system from the author in the future possibly. ***[Yes! -Dilwyn]***
Incidentally, the Format command doesn't just format floppy disks, anything you can do with a FORMAT command in BASIC seems to be possible - add *D or *H to a format name to force double or high density, ramdisk capacity can be specified. As the program is written in QLiberator compiled BASIC it's probably reasonable to assume that it uses a compiled format command anyway.
It took me some time to master this program and my initial impressions were that it's a complex program saved by a well designed and pretty intuitive user interface which allowed me to use it quickly at a simple level and get used to the more advanced features gradually. If you are used to QPAC2's files menu for example, most of it will be pretty familiar but there are enough slight differences to trip you up from time to time, sadly. The on-screen layout borders on being cluttered and intimidating at first, but persevere and you fairly rapidly get used to it. The facility to take advantage of File Info 2 without relying on it is quite useful. It comes with a fairly well written manual, although it's a bit long and heavy reading in places. I would have preferred a different name for the Trash can, as it clashes with the name of a similar but not compatible system on Qubides.
I can't really think of much I would add to this program without over-complicating it, as it seems packed to bursting point with facilities as it is. Perhaps a Backup command for making date-dependent backups, or a Tree command to allow whole sub-directories to be copied, or for file searching to search through sub-directories so that whole drives could be searched in one go.
Before suggesting additions, though, I should point out that this program is already 161 KB long. Compare that to just 38 KB for Qpac 2 for example. Qpac 2 is written in machine code though, while Qtrans is compiled basic. As most QL systems nowadays seem to have 2 whole megabytes or more of memory (apart from trump card or older systems) and when using other computers we are well used to programs which are huge by comparison, I suppose I shouldn't risk sounding like I'm harking back to the days when 64 KB was a huge memory on a computer!
As it stands, this program is quite worth getting, especially as it's (currently anyway) free! I downloaded it from the author's page
www.tesco.net/dilwyn.jones/freeware.html
as a zipped file. It has a few niggles and suffers from over-complexity and too many options in some cases, and takes a little getting used to (especially the slight differences to the QPAC2 way of doing things) at first, but put in the effort and if you need a good file handling program I'm sure you'll like it eventually. It needs pointer environment and Toolkit 2 but those are pretty well standard issue these days for most QL systems. And above all, as it's touted as being part of his Launchpad system (often mentioned but little seen so far apart from a few screen dumps in QL Today) Qtrans bodes well if representative of the quality of that system when it comes out!

QUANTA

Independent QL Users Group

Worldwide Membership is by subscription
only, and offers the following benefits:
Monthly Newsletter - up to 40 pages
Massive Software Library - All Free !
Free Helpline and Workshops
Regional Sub-Groups. One near you?
Advice on Software and Hardware problems
Subscription just £14 for UK members
Overseas subscription £17

Barclaycard: Visa: Access: Mastercard

Now in our Nineteenth Year

Further details from the Membership Secretary

John Gilpin, 181, Urmston Lane
Stretford, Manchester, M32 9EH
Tel. +44(0)1618 652872

or

Visit the Quanta Web Site

http://www.quanta.org.uk

Email: membership@quanta.org.uk

QLTdis - part 9

Norman Dunbar

We are getting close to the end now. This installment finishes off the last instruction decoder - from here on all we have to do is test, test and test again, probably followed by some fixing of existing code and then more testing.

After that, we need to go back over the code and see what we need to change to allow printing to an output file as well as the screen. I'm not looking forward to this on the grounds of not designing the program properly way back at the start - I've been a bit rushed (as you can probably tell by the numerous corrections I've had to do) and I did not design this program carefully enough - in my opinion.

Talking of code corrections, what better way to start ....

Code corrections

Oh hum, here we go again!

In the 'addr_reg' sub-routine the second line of code makes a branch to 'dr_exit', this should be to 'ar_exit' - although it doesn't make all that much difference really!

In the 'reg_list' sub-routine, change the following code:

```
rl_531  bsr     slash       ; Add a spare slash
        bne.s   rl_650      ; Then remove it again
```

to the following:

```
rl_531  bsr     slash       ; Add a spare slash
        bra.s   rl_650      ; Then remove it again
```

The branch is of course not conditional on the zero flag being clear, we always want to branch.

Still in the 'reg_list' sub-routine, the following code just above label 'rl_572' needs changing from this:

```
        move.b  #'/',d4     ; Assume more registers. D4 = '-Dn/' or '-An/'
        bra.s   rl_650      ; Check if we are done yet.
```

to this:

```
        move.b  #'/',d4     ; Assume more registers. D4 = '-Dn/' or '-An/'
        bsr     str_add_l   ; Update the buffer
        bra.s   rl_650      ; Check if we are done yet.
```

There is no point going through all the motions of working out a register list if we don't add it into the buffer when we are finished!

Similarly, the code just above label 'rl_650' also needs changing from this:

```
        lsl.l   #8,d4       ; D4 = '-Dn ' or '-An '
        move.b  #'/',d4     ; D4 = '-Dn/' or '-An/'
```

to the following:

```
        lsl.l   #8,d4       ; D4 = '-Dn ' or '-An '
        move.b  #'/',d4     ; D4 = '-Dn/' or '-An/'
        bsr     str_add_l   ; Update the buffer
```

On with the code

We only have one more instruction to decode, the MOVEM one. This has given me some fun and games to get it decoded correctly. Most of the fun was to be had in the register list decoder, however, everything seems to be working fine now (fingers crossed) so here is the code for type 27.

```
*---------------------------------------------------------------------------
* TYPE 27 - the MOVEM instructions
*---------------------------------------------------------------------------
dtype_27    btst    #6,d7           ; Check the size specifier
            beq.s   t27_word        ; Clear = .W
            bsr     ell             ; Add '.L' size details
            moveq   #4,d5           ; Set the op-code size too
            bra.s   t27_all         ; Skip word stuff

t27_word    bsr     uu              ; Add '.W' size details
            moveq   #2,d5           ; Set the op-code size

t27_all     bsr     space           ; Add a space to the buffer
            move.w  (a6)+,-(A7)     ; Stack the register list word
            btst    #10,d7          ; Set = Mem->Reg
            beq.s   t27_r2m         ; Clear = Reg->Mem

t27_m2r     bsr     eff_addr        ; Effective address is extracted
            bsr     comma           ; Then a comma
            move.w  (a7)+,d2        ; Get the register list word
            ror.w   #8,d2           ; rotate the high byte -> low byte
            move.b  d2,d4           ; Copy the (old) high byte
            bsr     addr_reg        ; Extract the address registers
            ror.w   #8,d2           ; (old) low byte back again
            bsr     t27_slash       ; Add a slash if required
            move.b  d2,d4           ; Get the data register list
            bsr     data_reg        ; Extract them
            bra     p_hex           ; Finished with mem -> reg

t27_r2m     move.w  (a7)+,d2        ; Fetch the register list word
            andi.b  #$38,d0         ; Extract the mode bits
            cmpi.b  #$20,d0         ; Mode = '100' = pre-decrement
            bne.s   t27_notpd       ; Not pre-decrement
            bsr     swap_d2         ; Reverse the bits in D2 for pre-dec

t27_notpd   move.b  d2,d4           ; Data registers in low byte
            bsr     data_reg        ; Extract them
            ror.w   #8,d2           ; Shift high byte to low byte
            move.b  d2,d4           ; Address register list
            bsr     t27_slash       ; Add a slash if required
            bsr     addr_reg        ; Extract address register list
            bsr     comma           ; Comma required
            bsr     eff_addr        ; And finally, the effective address
            bra     p_hex           ; Finished with reg -> mem

*---------------------------------------------------------------------------
* T27_SLASH - adds a slash if required
*
* D4.B = addr list mask
* D2.B = data register mask
*
* A slash will be required if both are non-zero.
*---------------------------------------------------------------------------
t27_slash   tst.b   d2              ; No slash if zero
            beq.s   t27_nosl2
            tst.b   d4              ; No slash if zero
            beq.s   t27_nosl2
            bsr     slash           ; Add a '/'
t27_nosl2   rts
```

So that's the end of the decoding. As I have been writing these routines I've been testing them with as much data as I possibly can. This way I try to fully exercise all routes though each routine - it can be quite a pain making up all the test code I can tell you.

One thing has become clear however, some instructions are quite large when decoded, especially some of the MOVE ‹ea› , ‹ea› ones take up a lot of room on a line.

Up until now, we've been printing the ASCII representation of the instructions bytes after the fully decoded instruction, starting at column 65. Well, in testing I found that column 65 was not far enough over and the ASCII code overwrote the last few characters of the decoded instruction - which is not a lot of help.

I could fix this by making the output window even wider and tabbing over to some other column rather than 65, however, as this could be a problem with other code I (or you) might write in future, I thought of a useful sub-routine that would tab to a given column if the current cursor position has not got there yet, but if it is already past the required position, would cause printing to start on the next line instead.

The following code should now be added to the end of UTILS_ASM:

```
*--------------------------------------------------------------------------------
* A routine to tab to column D1.W on the current output line, or to throw a
* new line if D1 › current tab position.
*--------------------------------------------------------------------------------
tab_enq     moveq   #sd_chenq,d0     ; Channel enquiry in character sizes
            moveq   #infinite,d3     ; Timout
            lea     tab_buffer,a1    ; 4 Word buffer for results
            move.l  a1,-(a7)         ; Stack the buffer address
            bsr.s   trap_3           ; Do it
            move.l  (a7)+,a1         ; Restore buffer address
            cmp.w   4(a1),d1         ; Compare cursor pos with desired tab
            bhi.s   tab_to           ; D1 › current pos - just do the tab
            move.w  d1,-(a7)         ; Save the tab position required
            moveq   #linefeed,d1     ; We need a linefeed
            bsr.s   one_byte         ; Do it
            move.w  (a7)+,d1         ; Get the required tab again
            bra.s   tab_to           ; And tab to it

tab_buffer  ds.w    4                ; 4 words required.
```

The above code simply takes a look at the channel dimensions for the channel in A0.L (which is set up in our main disassembly loop) and fills a 4 word buffer with the dimensions of the channel plus the current column and line position in the channel.

If we know where the cursor is, we can test it to see if it is to the right of where we want it to be, and if so, simply tab over. If it is to the left of where we want to be, we throw a newline and then tab over.

Now that we have added this useful routine, we need to call it. Locate the following code in the file diss_asm at around line 165:

```
*--------------------------------------------------------------------------------
* Now we have printed the hex codes, tab to column 33 ready for the decoded
* instruction text from the output buffer.
*--------------------------------------------------------------------------------
            moveq   #33,d1           ; Column 33
            bsr     tab_to           ; Tab to column 33
            move.l  a5,a1            ; Output buffer
            bsr     prompt           ; Print it
            moveq   #65,d1           ; Column 65
            bsr     tab_to           ; Tab to column 65
```

and change the last line above to the following

```
            bsr     tab_enq          ; Tab to column 65
```

As you can see, we are now going to tab to column 65 if there is room on the current line and if not, we will tab to column 65 on the next line instead.

That's all for this issue, next time I'll hopefully have had time to do some more testing (I really need to exercise the MOVEM routines thouroughly) and maybe it will all just work. On the other hand, previous experience tells me that some changes will be required!

See you then.

List Keywords

Dilwyn Jones

This is a short BASIC program to display a list of extension keywords in a BASIC extensions file.
It is not perfect - as far as I know, there is no official documented way of doing this other than to install the file and study the list of extensions made available, or to study the assembler source file.

Often, all you need is a simple list of extensions in a toolkit or machine code extensions file to check if there's a clash of keyword names between toolkits, for example. Another requirement arose a few months ago when we began to document the list of keywords in all the toolkits we were aware of to assist software writers of new toolkits to avoid name clashes.

The approach I took was as follows. The QL Technical guides from Sinclair and others document the format used to create an extensions file and it usually takes the form of 10 or 12 bytes of machine code to point to the definition table, something like this:

```
         LEA       table,A1
         MOVE.W    $110,A2
         JMP       (A2)

table    word      approximate number of procedures
* for each procedure:                        }
         word      pointer to routine-here   } omitted if no
         byte      length of name of procedure } procedures
         align to even address               }

         word      0 = end of procedures

         word      approximate number of functions
* for each function:                         }
         word      pointer to routine-here   } omitted if no
         byte      length of name of function } functions
         align to even address               }

         word      0 = end of functions
```

The above listing is a simplified breakdown of the information needed by my program to search for keywords defined. It starts by finding the definition table by finding the effective address loaded into register A1 prior to calling vectored routine $110 (normally referred to as bp.init in QDOS terminology, or sb.inipr in SMSQ terminology) to link in the new keywords.

Where this routine has been placed at the beginning of the extension file, the program generally copes OK, indeed most extensions files have the table 10 or 12 bytes into the file, hence the reason why this program searches the first 12 bytes for the word value of hexadecimal 43FA (17402 in decimal), the machine code values representing the LEA ....A1 instruction. If this is embedded further within the file, you will need to change the range of the 'a' loop in line 210 of the program to search further than 10 or 12 bytes inward.

Once the LEA instruction has been found, the offset to the definition table is calculated and the table examined. First, the number of procedures is extracted and this is only an "approximate" value since the value is actually defined as (total number of characters in names+number of procedures or functions+7)/8 if the average length of the names exceeds 7.

The REPEAT loop (called 'loop') steps through the names of the procedures one by one until the word value 0 is found, which signifies the end of the list of names of procedures. The process is then repeated for the list of functions.

This routine is a bit of a quick and dirty method of extracting this information but seems to work reasonably well in practice, although you may have to tweak line 210's search range a little for best results. The danger with searching too far into a file of course is that you may find another LEA...A1 instruction which is a part of the main code, not the definition.

```
100 REMark List keywords in rext file, by Dilwyn Jones
110 :
120 CLS : CLS #0
130 INPUT #0,'Filename of extensions file > ';ip$
140 fl = FLEN(\ip$)
150 base=ALCHP(fl)
160 LBYTES ip$,base
170 offset=-1
180 REMark search for LEA effective_address,A1 opcode
190 REMark change the 0 TO 10 to search however far you think is
200 REMark required through the rext file
210 FOR a = 0 TO 10 STEP 2
220   IF PEEK_W(base+a) = HEX('43FA') THEN
230     offset = a+2+PEEK_W(base+a+2) : REMark table here
240     EXIT a
250   END IF
260 END FOR a
270 IF offset = -1 THEN STOP : REMark not found
280 approx_procs = PEEK_W(base+offset)
290 :
300 PRINT'Approx. no. of procs = ';approx_procs
310 offset=offset+2
320 REPeat loop
330   pntr = PEEK_W(base+offset) : offset = offset+2
340   IF pntr = 0 THEN EXIT loop : REMark end of procs
350   namelen=PEEK(base+offset)
360   nme$=''
370   FOR a = 1 TO namelen
380     nme$ = nme$&CHR$(PEEK(base+offset+a))
390   END FOR a
400   PRINT !!nme$
410   REMark point to next name, but must round up to even
420   offset = offset+1+namelen+((namelen MOD 2)=0)
430 END REPeat loop
440 :
450 approx_fns = PEEK_W(base+offset)
460 PRINT\'Approx. no. of fns=';approx_fns
470 offset=offset+2
480 REPeat loop
490   pntr=PEEK_W(base+offset) : offset=offset+2
500   IF pntr = 0 THEN EXIT loop : REMark end of procs
510   namelen=PEEK(base+offset)
520   nme$=''
530   FOR a = 1 TO namelen
540     nme$ = nme$&CHR$(PEEK(base+offset+a))
550   END FOR a
560   PRINT !!nme$
570   REMark point to next name, but must round up to even
580   offset = offset+1+namelen+((namelen MOD 2)=0)
590 END REPeat loop
600 PRINT#0,'Program finished'
610 RECHP base
```

New functionalities in SMSQ/E - Part 1

Wolfgang Lenerz

SMSQ/E v.3.00 is not out quite yet.

Here, however, is a sneak preview of the new functionalities that will be contained in it. This essentially is a rehash of the documentation, prepared for the programmer, that accompanies the new version. It was drafted by Marcel Kingus (who, after all, programmed nearly all of this stuff), and just put into shape for magazine publication by me.

JOCHEN MERZ SOFTWARE

Im stillen Winkel 12 D-47169 Duisburg
Tel. 0203 502011 Fax 0203 502012
http://smsq.j-m-s.com

We hope to have all the updates ready end of April!

Well, this is the last page I'm preparing for this magazine, and I have waited until I actually had to print the master before I bring it to the printer.
Unfortunately, I do not know yet when the new SMSQ/E will be released, and as you can read somewhere else in this magazine, the new features require the new Window Manager and the new SMSQ/E.

QD is ready and waiting for its release.
QMENU is ready as well, but I think I'm going to rename it. There seems to be confusion between the Menu Extension (called by some people QMENU, but not by me) and QMENU, which is the whole product (Menu Extension, examples, programming instructions). I think QMENU is not a bad name for the Menu Extension on its own, so I'll think of something for the product.
The documentation needs to be updated anyway, so there's some work for me until the release date.
QSpread is more or less ready - a few missing icons from Phoebus arrived today.

I hope for sure that everything will be ready for the Hove Show, so expect upgrade possibilities by the end of April, beginning of May.

As the software has undergone major visual changes and the old colourways are not supported anymore, thanks to the much better 4 colour schemes provided by the new WMAN, it may be a good idea to keep the contents of the old disks.
I've decided that what I'll do for upgrades is: return the old disk with the contents unmodified (just as proof of purchase, as usual) and send out a new disk as well with the new programs.
In addition, we will only provide manuals in English from now on. Maintaining two languages is a vast amount of work - and I have to start thinking at least a little bit economical somewhere. You will, however, still not only find English but also German (and sometimes French) versions of the products on the disk, so that you can continue using what you're used too.

You can check the website - I'll update it when the updates are available - promised!

```
Vector $7C                                           WM.SETSP

     Set system palette entries

Call parameters                      Return parameters

D1.w start index                     D1   preserved
D2.w number of elements              D2   preserved
                                     D3+  all preserved

A0                                   A0   preserved
A1   pointer to palette entries / 0  A1   preserved
A2                                   A2   preserved
A3                                   A3   preserved
A4                                   A4   preserved
A5   not used by any routine
A6   not used by any routine

Error returns:
     IPAR    Illegal index number / invalid number of elements
```

Set the entries of the system palette to the values in the buffer, beginning with the index in D1 (counting from 0) and ending with the index D1 + D2 - 1.

If A1 = 0 then the entries are taken out of the default table. Otherwise the buffer must hold an array of words with the colour values of the different items. The colour format is the standard WMAN colour format as described elsewhere (notably earlier in QL Today).

```
Vector $80                                                   WM.GETSP

     Read system palette entries

Call parameters                      Return parameters

D1.w start index                     D1   preserved
D2.w number of elements / -1         D2.w preserved / item count
                                     D3+  all preserved

A0                                   A0   preserved
A1   pointer to entry buffer         A1   preserved
A2                                   A2   preserved
A3                                   A3   preserved
A4                                   A4   preserved
A5   not used by any routine
A6   not used by any routine

Error returns:
     IPAR    Illegal index number / invalid number of elements
```

Copies entries of the system palette into the given buffer, beginning with the index in D1 (counting from 0) and ending with the index D1 + D2 - 1. The buffer must be big enough to hold all requested entries.

If D1 is given as -1 the function just returns the number of items held in the system palette. This can increase when more items get defined in new WMAN version. This is guaranteed to be below 256.

```
Vector $84                                       WM.TRAP3

     Trap #3 replacement that handles WMAN colour codes

Call parameters                       Return parameters

D0.l function code                    D0   error code
D1.w colour code                      D1   preserved
D2+  parameter                        D2+  result according to trap

A0.l channel id                       A0   preserved
A1+  parameter                        A1+  result according to trap

Error returns:
     same as original traps
```

This is a drop-in replacement for a "trap #3" call. D0 reacts to any of the codes iow.defb, iow.defw, iow.spap, iow.sstr, iow.sink and iow.blok. Those routines are exchanged by some that can handle the extended WMAN colour codes. Other function codes are directly passed to an ordinary "trap #3" call. The condiditon codes are guaranteed to be set properly according to D0, for all traps.

```
Vector $88                                            WM.OPW

     Emulate OPW.WIND, OPW.CON and OPW.SCR vectored routines

Call parameters                       Return parameters

D0.l OPW.WIND, OPW.CON or OPW.SCR     D0    error code
D1                                    D1    smashed
D2                                    D2    smashed
D3                                    D3    smashed

A0.l ptr to name (OPW.WIND only)      A0.l  channel ID
A1.l ptr to parameter block           A1    smashed
A2                                    A2    smashed
A3                                    A3    smashed

Error returns:
     same as original functions
```

This is a replacement for the OPW.WIND, OPW.CON and OPW.SCR vectored routines. Howeveer, in contrast to the originals the paramater block pointed to by A1 is in words instead of bytes:

$00	border colour (word)
$02	border width (word)
$04	paper/strip colour (word)
$06	ink colour (word)

OPW.CON and OPW.SCR define the window using an additional block of four words:

$08	width (word)
$0A	height (word)
$0C	X-origin (word)
$0E	Y-origin (word)

```
Vector $8C                                        WM.SSCLR

     Set single colour pattern

Call parameters                      Return parameters

D0                                   D0   0
D1.w colour number                   D1   preserved
                                     D2+  all preserved

A1.l ptr to window status area       A1   preserved
A2.l ptr to pattern space            A2   preserved
A3                                   A3   preserved
A4                                   A4   preserved
A5   not used by any routine
A6   not used by any routine
```

Returns a pattern that is filled with the given colour. The space pointed to by A1 must hold at least $60 bytes. Does not work for stippled colours.

```
Vector $90                                   WM.JBPAL

     Set system palette number of job

Call parameters                      Return parameters

D1.l job ID / -1                     D1   preserved
D2                                   D2   preserved
D3.w palette number / -1             D3+  all preserved

A0                                   A0   preserved
A1   ptr to job palette or 0 (D3=-1) A1   preserved
A2                                   A2   preserved
A3                                   A3   preserved
A4                                   A4   preserved
A5   not used by any routine
A6   not used by any routine

Error returns:
     IJOB    Invalid job ID
```

Sets the active system palette for the given job. If D1 is -1 then the current job will be used. D3 can be supplied as -1 which can be used to give the job its very own palette. In this case a pointer to the palette can be supplied in A1. Attention: the contents of this area is not copied, it is used directly and must remain there as long as the job uses this palette! If A1 is supplied as 0 the palette pointer will not be touched.

II - Sprites

Sprites have undergone a serious lifting in this version of SMSQ/E. Largely thanks to Jérôme Grimbert there are many more types of sprites, nearly one per GD2 mode (eg. QL, 24 bit, QPC native, Qx0 native etc)....

TF Services

Compswitch

A UK 4-way trailing socket designed to switch off computer peripherals automatically when the computer is switched off, or (in the case of an ATX computer) when it auto-powers down. *Compswitch* has one control socket, and three switched sockets.
Can be used with lights/hifi/monitors—ie a QL monitor can be used as a switch control.

Cost £24

superHermes

A major hardware upgrade for the QL
All Hermes features (working serl/2 at 19200, independent baud rates/de-bounced keyboard/ keyclick) IBM AT kbd I/F // HIGH SPEED RS232 at 57600// serial mouse port and 2 other RS232 inputs// 3 I/O lines // EEPROM

Cost (including manual/software) ....**£90** (£92/£93)
IBM AT UK layout Keyboard...............**£11** (£13/£15)
Serial mouse..**£8** (£8.50/£9)
Capslock/scrollock LED ....**£1** (£1.50/£1.50)
Keyboard or mouse lead ....**£3** (£3.50/£3.50)
High speed serial (ser3) lead.............**£4** (£4.50/£4.50)

Hermes available for £25 (£26/£27) Working ser1/2 and independent input, debounced keyboard.

SuperHermes LITE: All Hermes features (see above) + an IBM AT keyboard interface only.
Cost (incl keyboard lead) ....**£53** (£54/£55)

QL REPAIRS (UK only)

Fixed price for unmodified QLs, excl microdrives. QLs tested with Thorn-EMI rig and ROM software.

£27 incl 6 month guarantee

Minerva

The ORIGINAL system operating system upgrade

OTHER FEATURES COMMON TO ALL VERSIONS

DEBUGGED operating system/ autoboot on reset of power failure/ Multiple Basic/ faster scheduler- graphics (within 10% of lightning) - string handling/ WHEN ERROR/ 2nd screen/ TRACE/ non-English keyboard drivers/ "warm" fast reset. V1.97 with split OUTPUT baud rates (+ Hermes) & built in Multibasic.

First upgrade free. Otherwise send £3 (+£5 for manual if requd). Send disk plus SAE or two IRCs

MKI...£40 (£41/£43) **MKII...£65** (£66/£67)

MINERVA RTC (MKII) + battery for 256 bytes ram. CRASHPROOF clock & I^{2}C bus for interfacing. Can autoboot from battery backed ram. Quick start-up.

QL RomDisq

Up to 8 mbyte of flash memory for the QL
A small plug in circuit for the QL's ROM port (or Aurora) giving 2, 4 or 8 mbytes of permanent storage - it can be thought of as a portable hard disk on a card, and reads at some 2 mbytes per second.
Think of it - you could fully boot an expanded QL, including all drivers/SMSQ etc off **RomDisq** at hard disk speed with only a memory expansion needed.

2 mbytes RomDisq............**£39** (£40/£41)
4mbytes RomDisq..............**£65**(£66/£67)
8 mbytes RomDisq..........**£98** (£99/£100)
Aurora adaptor......................**£3** (£3.50/£4)

MPLANE

A low profile powered backplane with ROM port

A three expansion backplane with ROM port included for RomDisq etc. Aurora can be fitted in notebook case and powered off single 5V rail - contact QBranch for details. Two boards (eg Aurora and Gold Card/Super Gold Card/Goldfire fixed to base. Suitable for Aurora (ROM accessible from outside) & QL motherboard in tower case. Specify ROM facing IN towards boards, or OUT towards back of case.

Cost..**£34** (£35/£36)

I2C INTERFACES

Connects to Minerva MKII and any Philips I^{2}C bus

Power Driver Interface 16 I/O lines with 12 of these used to control 8 current carrying outputs (source and sink capable)
2 amp (for 8 relays, small motors)....................**£40** (£43/£44)
4 amp total (for motors etc).......................**£45** (£48/£50)
Relays (8 3a 12v 2-way mains relays (needs 2a power driver)..**£25** (£28/£29)
Parallel Interface Gives 16 input/output lines. Can be used wherever logic signals are required...........**£25** (£27/£28)
Analogue Interface Gives eight 8 bit analogue to digital inputs (ADC) and two 8 bit digital to analogue outputs (DAC). Used for temp measurements, sound sampling (to 5 KHz), x/y plotting.....................................**£30** (£31/£32)
Temp probe (-40°C to +125°C)...............**£10** (£10.50/£11)
Connector for four temp probes..............**£10** (£10.50/£11)
Data sheets...**£2** (£2.50/£3)
Control software & manual (for all I/F).........**£2** (£2.50/£3)

QL SPARES

Keyboard membrane .... **no longer on sale**
1377 PAL ....**£3** (£3.50/£4)
Circuit diagrams..**£3** (£3.50/£4)
68008 cpu or 8049 IPC.....................................**£8** (£8.50/£9)
8301/8302 or JM ROM or serial lead..........**£10** (£10.50/£11)
Power supply (sea mail overseas)...................**£12** (£19/£23)
Other components (sockets etc) also available

Prices include postage and packing (Airmail where applicable) Prices are: UK (Europe /Rest of world). Payment by cheque drawn on bank with UK address,/postal order or CASH! I can no longer accept card payments as UK only does PDQ transaction. SAE or IRC for full list and details 22 Feb 03

29 Longfield Road, TRING, Herts, HP23 4DG
Tel: +44 (0) 1442-828254 Fax/BBS: +44 (0) 1442-828255
tony@firshman.co.uk http://www.firshman.co.uk

A - Sprite definition

The sprite definition has been amended/extended with respect to the original definition. It is now as follows:

```
$00  byte  sprite mode
$01  byte  colour mode / system sprite number
$02  byte  dynamic sprite version number
$03  byte  cache control version number
$04  word  X size
$06  word  Y size
$08  word  X offset
$0a  word  Y offset
$0c  long  relative pointer to colour pattern
$10  long  relative pointer to mask/alpha channel
$14  long  relative pointer to next object

$18  long  $53705274 ('SpRt')  **- optional -**
$1C  long  rel. pointer to "sprite pointer block" **- optional -**
```

B - Sprite mode

The Sprite mode can be any of the following:

```
0       system sprite
1       traditional QL colour sprite
2       GD2 colour sprite
```

<u>System sprite:</u>
When the sprite mode is 0 for system sprite then the second byte is the number of the sprite. ALL other values are ignored in that case, i.e. a system sprite reference is only 2 bytes long.

<u>QL colour sprite:</u>

```
0       mode 4
1       mode 8
```

<u>GD2 colour sprite:</u>

```
0       1 bit black&white
3       1 bit palette mapped
4       2 bit fixed gr palette
7       2 bit palette mapped
8       4 bit fixed irgb palette
15      4 bit palette mapped
16      8 bit fixed palette (equals Aurora palette)
31      8 bit palette mapped
32      16 bit QPC/QXL %gggbbbbbrrrrrggg format
33      16 bit Q40 %gggggrrrrrbbbbbw format
64      32 bit $RRGGBB00 format
```

C - Alpha channel:

In GD2 modes, when the mask begins with the bytes 'AlPh' the mask is considered to be an alpha channel. An alpha channel allows gradual mixes between the background and the sprite pattern. Every pixel is represented by exactly one byte. 0 means the pixel is completely transparent, 255 means the pixel is completely opaque. Values in between determine the degree of mixing of background and foreground. Alpha channel information is not padded at the end of each line. There's one byte for every pixel and nothing more.

D - RLE compression:

Both pattern and mask/alpha channel can be compressed using a simple RLE (run length encoding) algorithm. This is usefull with data that is largely homogene, which is often the case with masks. Compressed data starts with the bytes 'RLEx', with 'x' being either 1, 2 or 4. This is the item size the algorithm is working with. 8 bit RLE compression of 32 bit data wouldn't yield good results, therefore the algoritm can also work on 16 bit or 32 bit data. After the ID there's one long word containing the size of the data in uncompressed form. After that the compressed data itself is following.

The compressed data always consists of one byte and one or more items. If the leading byte is in the range of 0<=x<128 then x+1 uncompressed items are following. Otherwise only one item is following, which represents 257-x times the same item in the uncompressed data.

Note: the alpha channel ID must not be encrypted with the rest of the data.

E - New sprite block definition

The object drawing routines have been amended so as to allow different sprites to be drawn in a loose item, depending on the status of that item.

In such a case, it is up to the application to supply the different sprites.

To keep things compatible with older versions of WMAN, this has been handled by extending the sprite definition of the first sprite, by the two optional parameters shown above.

As usual, a sprite is supplied as an object for the loose item in question.

However, for this sprite (the "original" sprite), the pointer to the next sprite is followed by a long word $53705274 ('SpRt') and then by a long relative pointer to a "sprite pointer block".

The sprite pointer block is just a block of 5 longword pointers:

* pointer to sprite if item is available
* pointer to sprite if item is available AND is the current item
* pointer to sprite if item is selected
* pointer to sprite if item is selected AND is the current item
* pointer to sprite if item is unavailable

In all cases, these are longword relative pointers.

All but the first pointer may be 0. The first pointer (item available) MUST exist and MUST point to a real sprite.

0 pointers are handled as follows:
- For available items:
  * The pointer to the available item sprite MUST exist.
  * If no pointer to an available and current item sprite exists, then the available item sprite is taken instead
- For selected items:
  * If no pointer to a selected item exists, then the pointer to the selected item AND current item is ALSO ignored. The avilable item sprite is taken instead for both.
  * If no pointer to a selected AND current item sprite exists, then the selected item sprite is taken instead.
- For unavailable items:
  * the available item sprite is used.

It is allowed, but not necessary, for any of these pointers including the first pointer (available item) to point back to the original sprite, which will then be drawn as a normal sprite.

This allows three cases:

1 - The original sprite can be an ordinary QL mode sprite, which will be drawn normally by older versions of WMAN. The newer versions of WMAN will use the extended format.

2 - The original sprite could be a simple empty shell, with just the relevant data (long word $53705274 ('SpRt') and pointer to sprite pointer block) set.

3 - The original sprite could be a normal QL or 24 bit mode sprite which will be used by an item in any of its statusses.

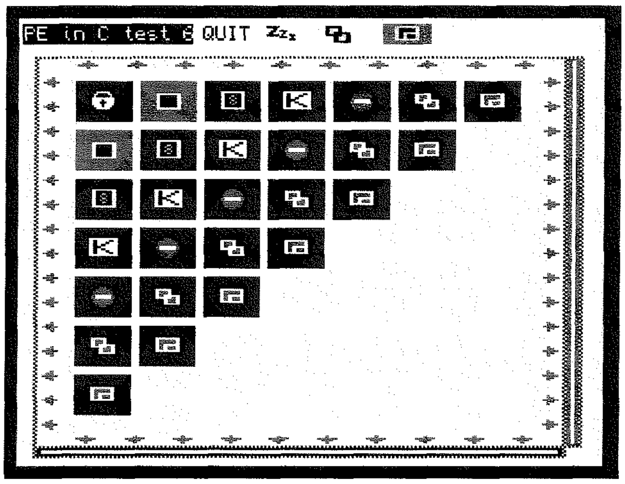
Alternative 1 above will ensure that your software remains compatible with older versions of WMAN.

Next issue, the new colour format and the system pallette will be explained.

A short Visit of XMenu - Part 6

Jérôme Grimbert

Splitting a sub-window



Transforming the application window in order to be able to split it is rather easy as we are going to see:

```
--- CPE5_C
+++ CPE6_C
@@ -13,13 +13,14 @@
 char _conname[] = "con_2x1a0x0";
  /* mask startup problems, for old one */
 char *_endmsg  = NULL;
  /* and stop when I say */

-char _PROG_NAME[] = "PE in C tutorial 5";
+char _PROG_NAME[] = "PE in C tutorial 6";
 static QD_TEXTI(quit,"QUIT");
-static QD_TEXTI(title,"PE in C test 5");
+static QD_TEXTI(title,"PE in C test 6");
```

```
  static long MY_MENU_DRAW(struct WM_wwork *wwk, struct
  WM_menw *mw)
  {
+  wm_index(wwk,(struct WM_swdef *)mw);
   return wm_mdraw(wwk,(struct WM_swdef *)mw,0);
  }
```

Nothing really important above, just a change of label (increasing the 5 in a 6).

But because we want scroll bars and all the things, we need to add a call to wm_index() in the drawing routine.

```
@@ -102,8 +103,13 @@
    struct WM_appl *al;
    struct WM_mobj *menuobj;
    struct WM_rowl *row;
+   struct WM_pwcb *pwcy;
+   struct WM_pwcb *pwcx;
    char *mstt;

+   pwcy = (struct WM_pwcb *)malloc(sizeof(struct
     WM_pwcb)*3);
+   pwcx = (struct WM_pwcb *)malloc(sizeof(struct
     WM_pwcb)*3);
+
    info_list=(struct WM_info *)malloc(sizeof(struct
     WM_info)*2);
    info_list[0].xsize=14*6;
    info_list[0].ysize=10;
```

We are going to split the window both ways. We need some memory to store the information about each section.

As we intend to limit the number of splits to 3 in each direction, we only need the memory for three structures in each orientation.

```
@@ -205,8 +211,8 @@
 }
```

```
      aw = (struct WM_menw *) malloc(sizeof(struct
       WM_menw));
-     aw->xsize=12*20;
-     aw->ysize=180;
+     aw->xsize=13*20;
+     aw->ysize=190;
      aw->xorg=10;
      aw->yorg=20;
      aw->flag=-32768;
```

Notice that arrows are drawn inside the area, whereas the scrollbars are drawn outside.

This means that we give more room for the arrow here. It is not mandatory, but it helps to keep the same starting view.

```
@@ -216,9 +222,9 @@
      aw->pspr=&wm_sprite_hand;
      aw->draw=&menu_draw;
      aw->hit=wm__mhit;
-     aw->ctrl=NULL;
-     aw->nxsc=0;
-     aw->nysc=0;
+     aw->ctrl=wm__pnsc;
+     aw->nxsc=3;
+     aw->nysc=3;
      aw->skey=K_TAB;
      aw->ncol = 7;
      aw->nrow = 7;
```

Now, pay attention. We need to have a control routine which will handle the scroll/split for us.

The Window Manager comes with one, so let's use it instead of doing it by hand.

Also, we need to bound the number of section for each orientation.

Whenever the nxsc or nysc is 0, there is no scroll bar, no arrow. At 1, you can scroll but not split. The value is the maximal number of sections that the programmer allows.

```
@@ -228,9 +234,17 @@
      aw->xs.c._spce = -32;
      aw->ys.c._size = -20;
      aw->ys.c._spce = -24;
+     pwcy->nsec = 1;
+     pwcx->nsec = 1;
+     pwcy->s[0].stt = 0;
+     pwcx->s[0].stt = 0;
+     pwcy->s[0].pos = 0;
+     pwcx->s[0].pos = 0;
+     pwcy->s[0].siz = (aw->ysize - WW_YARROW *2)/24;
+     pwcx->s[0].siz = (aw->xsize - WW_XARROW *2)/32;
      aw->xind = 0;
      aw->yind = 0;
-     aw->curw = 2;
+     aw->curw = 1;
      aw->curc = 5;
      aw->uback = 0;
      aw->uink = 7;
```

We need to indicate that we want to start with one section. And that section must start at the first pixel (.pos), as the first row/column (.stt) and the number of row or columns that we can fit in the initial display (.siz).

JUST WORDS!

Ho, Ho, Hove!

JUST WORDS! hopes to be at HOVE with a host of NEW GOODIES for your delight:

VOCABULARY DATABASE - £5 or €7,50
5,000 words in English, German, French, Dutch and Japanese.

SCRABBLE LISTS - £1 or €1,50
Two lists of Scrabble approved words.

QTYP ENGLISH DICTIONARY - 194,000 words - £1 or €1,50
Mixed UK and USA spellings

QTYP ITALIAN DICTIONARY - 80,000 words - £1 or €1,50

QTYP DUTCH DICTIONARY - 180,000 words - £1 or €1,50
Corrected for 1995 spelling revision.

AND MAYBE, QTYP GERMAN DICTIONARY - 140,000+ words - old spellings.

Geoff Wicks, 56 Peveril Crescent, West Hallam, Derbyshire DE7 6ND, U.K.
Tel: +44 (0)115 - 930 3713 *email: gwicks@beeb.net*
Web: http://members.lycos.co.uk/geoffwicks/justwords.htm

```
@@ -238,33 +250,32 @@
     aw->aink = 6;
     aw->sback = 2;
     aw->sink= 5;
-    aw->xoff = 4;
-    aw->yoff = 4;
-    aw->ypwcb=NULL;
-    aw->xpwcb=NULL;
+    aw->xoff = 4 + WW_XARROW;
+    aw->yoff = 4 + WW_YARROW;
+    aw->xpwcb=pwcx;
     aw->xinsz=0;
     aw->xinsp=0;
     aw->xiciw=0;
-    aw->xicic=0;
-    aw->xiback=0;
-    aw->xiink=0;
+    aw->xicic=1;
+    aw->xiback=2;
+    aw->xiink=3;
     aw->xiblob=NULL;
     aw->xipatt=NULL;
-    aw->xpsac=0;
-    aw->xpsbc=0;
+    aw->xpsac=2;
+    aw->xpsbc=4;
     aw->xpssc=0;
-    aw->ypwcb=NULL;
+    aw->ypwcb=pwcy;
     aw->yinsz=0;
     aw->yinsp=0;
     aw->yiciw=0;
-    aw->yicic=0;
+    aw->yicic=1;
     aw->yiback=0;
-    aw->yiink=0;
+    aw->yiink=5;
     aw->yiblob=NULL;
     aw->yipatt=NULL;
-    aw->ypsac=0;
+    aw->ypsac=4;
     aw->ypsbc=0;
-    aw->ypssc=0;
+    aw->ypssc=2;

     al = (struct WM_appl *) malloc(2*sizeof(struct
      WM_appw *));
     al[0].pappw= (struct WM_appw *)aw;
```

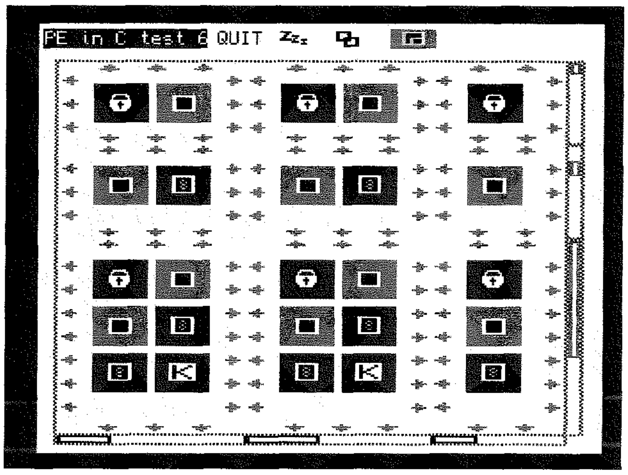
Above, we put the pointer of the section-control area in the application window (so that the control routine will find it), and we add some pixels for the offset, so that the arrows can be drawn.

We also change some attributes so that we can see the arrows and scrollbars.

```
@@ -282,8 +293,8 @@
     result->pulld=0;

     result->splst=al;
-    result->xsize=20*13;
-    result->ysize=20+190;
+    result->xsize=20*14+10;
+    result->ysize=40+180;
     result->xorg=20; /* initial position of mouse */
     result->yorg=8;
```

And finally, we extend the main window, because, and this is important this time, the scrollbars are outside of the application window.

If the application window was tight-fitted in the main window, we would have a problem to place the scrollbars.



This is the kind of sillyness the user can now make with our little application.

That's enough for this series of articles and I want to thank the people which positively answers to the first articles by asking more things (such as splitable window) where I only intended at first a small tutorial. Of course, if there are additional questions (I know my text are usually a little short for the explanation), I will be glad to answer them in these pages (and more knowledgeable people might also want to correct me or provide more details).

Jérôme Grimbert, **jgrimbert@free.fr**

Good to read that there has been actually response to this series - Dear readers, always consider that authors put a lot of effort into writing articles and probably enjoy nothing as much as feedback.
Jérôme has provided us with the full listing in one go. We don't want to print it in the current issue otherwise we end up with lots of long listings - if there is demand, we will print it in the next issue, of course! Let us know:
editor@qltoday.com

Have you renewed your subscription yet?

DISPLAY CODE Update

Dilwyn Jones

Way back in Volume 2 of QL Today I presented an assembler listing for some basic extensions designed to help BASIC programmers to handle high resolution screens. Although such extensions and facilities were provided by SBASIC on platforms using SMSQ/E, for example, these extensions gave a cross-platform means of extracting the relevant information from the system, enabling programmers to write programs in such a way that they could be used on older systems which did not support these facilities, or only partially supported them.

Recently, I have updated these extensions, adding a few new extensions aimed at extracting information regarding pointer environment and GD2 (the "colour drivers").

This is now a small suite of 14 basic extensions written in assembler designed to help SuperBASIC or SBASIC programmers cope with extended display modes on more recent hardware and emulators such as Aurora, QPC and QXL. It also provides a set of extensions to check for presence of pointer environment, window manager, GD2 (the so-called "colour drivers") and to check the version numbers of the pointer interface and operating system.

These extensions will work on SMSQ and QDOS, providing a means of consistently returning the required information, allowing programs a means of working on all systems. Graphical applications often need to write direct to memory, or at least to know the screen size and location details. That's the sort of information this code will allow you to extract from the system.

Over the years, many programs were written which were later found not to work on displays other than the original 512x256 QL screen. The forward-thinking designers of the QL had actually allowed for the possibility of larger screen sizes by including information in the system which was available to machine code programs, but not to SuperBASIC programmers. As the information about this was not readily documented and available or widely understood in those early days, many programs just assumed the original QL display and hence could not work properly when the size and location of the screen memory changed - I know, I wrote such programs myself! This set of extensions won't fix those early programs of course, but does give a simple means of extracting this information for SuperBASIC and SBASIC programmers so that new programs at least won't be guilty of the same sins. To be fair, with the benefit of hindsight it is easy to refer to those old programs as being sinful, though at the time the necessary information was neither readily accessible nor widely understood.

The assembler source listing is provided, along with a BASIC loader program which will generate a file called DISPLAY_CDE which can simply be LRESPRed to install the BASIC extensions. Alternatively, the file created by running the loader program can be installed with RESPR/LBYTES/CALL in the usual way:

```
base=RESPR(1094): LBYTES FLP1_DISPLAY_CDE,base:
CALL base
```

I hope that the names I have given the extensions don't clash with anything you already have installed on your system. If there is a clash, you'll need to either hack the names of the extensions in DISPLAY_CDE using your preferred editor, or reassemble the source code file DISPLAY_ASM after altering the names in the dc.b statements in the table.

The assembler listing includes a lot of repetition and could quite probably be condensed. However, I chose to write it in this way so that (1) readers can study the techniques I used to gather the system information and if required extract the necessary routines if just one of the extensions is required, and (2) as some of the methods I've used to gather information from the system are new and unfamiliar to me, it will make it easier to amend the code and publish corrections if significant improvements are suggested.

The Extensions

There are currently thirteen functions and one procedure. The ones from the original article are included of course, in case you do not have access to the original listings.

ADDRESS

```
LET adr = ADDRESS(#channel)
```

This function returns the base address of the display for the given window channel. Normally you'd use #0. For a 512x256. Normally, you'd only need this value if you were writing programs which wrote directly to the screen area, e.g. using LBYTES to load a screen direct to the video area of memory, using a command such as LBYTES filename,ADDRESS(#0) assuming the screen being loaded was the same size as the current display.

BYTES

```
LET bytes_per_line = BYTES(#channel)
```

The display is organised as a series of horizontal lines, with each line being a given number of bytes wide. In several display sizes, the exact width of these lines in bytes happens to be the number of pixels DIV 4, but this is a dangerous assumption to make - many programs made this assumption and fell over when the Aurora came along, as some of its display modes use a fixed line width, irrespective of the number of pixels on a line, meaning that some of the bytes used to store each line are actually unused. So if you are writing individual lines to the screen, as you would for video effects for example, you need to take account of how many bytes there are between the start of one line and the next. That's the purpose of this function - it tells you how many bytes lie between where one line starts and the next line starts. Users of version JM or earlier ROMs should note that this information is not available, as the area which holds this value is used for something else, so a version JM machine always assumes it has a 128 byte line width.

DMODE

```
LET display_mode = DMODE
```

Returns the mode number of the current display. This would usually be 0 for the 4 colour modes, and 8 for the 8 colour modes. As the routine uses the mt.inf system trap, it ought to handle the extended colour mode drivers, or monochrome modes on certain emulators (both cases untested at the time of writing). I am not sure what will happen if this function is used while one of the old mixed mode screen displays are used (there are extensions in Quanta library I think which allow part of the screen to be in MODE 4 and part in MODE 8, for example)

SYS_VAR

```
LET system_vars = SYS_VAR
```

Tells you at what address you can find the system variables. Now you can PEEK and POKE to your heart's content if you really need to!!!

The FLIM_n extensions return information about the maximum sizes or limits of a screen window size. As it uses the iop.flim trap, it means the information can't be extracted if this is not implemented on your system. But I think I'm right in assuming that if iop.flim is not on the system for whatever reason, the system can't use extended displays anyhow. If it can't get the required information, it assumes you're running a 512x256 QL screen rather than unhelpfully causing an error report. If you supply a primary channel window, the values returned will be the maximum possible size for that window (essentially the full display width and height), whereas if you supply a secondary channel number the values returned refer to the outline area for the primary. If you don't know what this means, supply the lowest window channel number opened, e.g. #0 for BASIC. The first two extensions return origin details, whereas the other two return the width and height details.

FLIM_X

```
LET x_origin = FLIM_X(#channel)
```

FLIM_Y

```
LET y_origin = FLIM_Y(#channel)
```

FLIM_W

```
LET wide = FLIM_W(#channel)
```

FLIM_H

```
LET high = FLIM_H(#channel)
```

OS_VER$

```
LET v$ = OS_VER$(#channel)
```

Checks the version number of QDOS or SMSQ, returning a 4 digit version string such as 1.23

PTR_ENV

```
LET ptr_present = PTR_ENV(#channel)
```

Returns 1 if pointer environment is present, or 0 if not.

WIN_MAN

```
LET wman_present = WIN_MAN(#channel)
```

Returns 1 if the Window Manager is present, or 0 if not. Unlikely to be useful, unless a QDOS user has only installed ptr_gen and failed to install WMAN.

PTRVER$

```
LET v$ = PTRVER$(#channel)
```

Returns the 4-digit version number of the pointer interface, or an empty or nul string if this cannot be found (e.g. no pointer interface present).

WMAVER$

```
LET v$ = WMAVER$(#channel)
```

Returns the 4-digit version number of the window manager or an empty or nul string if this cannot be found (e.g. no pointer environment present).

GD2

`LET gd2_present = GD2 (#channel)`

Returns 1 if the Graphics Device 2 is present, or 0 if not or unable to find this flag. For experts, the method of checking involves examining the PE linkage block at offset $128 (decimal 296) for the long word flag 'PTR2', the only method I know of to test for GD2 at present.

`MOVEMEM from_address, to_address, number_of_bytes`

This procedure lets you move the content of memory around. Simply tell it where to move from, where to move it to, and how many bytes. Negative values will cause errors. There is no check on overlaps etc so with care you can use this to fill memory areas by writing the first byte value with a POKE, for example, then moving this up to fill the required number of bytes. It always moves from lower addresses first - there is nothing particularly intelligent about this command in terms of working out the best way to move things. It is quite slow by comparison with similar commands in other toolkits.

You can type in the loader_bas program to generate the extension file without having to type in the full assembler listing - not even an assembler is required:

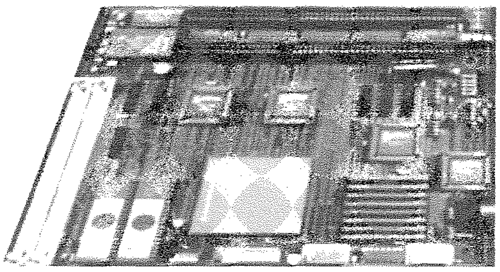
```
100 REMark generate display_cde extensions file
110 fl = 1094 : REMark length of code file
120 base = ALCHP(fl)
130 RESTORE
140 FOR a = 0 TO fl-1
150   READ byte : POKE base+a,byte
160 END FOR a
170 SBYTES ram1_display_cde,base,fl-1
180 :
190 DATA 67,250,0,8,52,120,1,16,78,210
200 DATA 0,1,0,148,7,77,79,86,69,77
210 DATA 69,77,0,0,0,14,0,198,7,65
220 DATA 68,68,82,69,83,83,1,0,5,66
230 DATA 89,84,69,83,1,76,5,68,77,79
240 DATA 68,69,1,100,7,83,89,83,95,86
250 DATA 65,82,1,136,6,70,76,73,77,95
260 DATA 88,0,1,134,6,70,76,73,77,95
270 DATA 89,0,1,100,6,70,76,73,77,95
280 DATA 87,0,1,98,6,70,76,73,77,95
290 DATA 72,0,1,190,7,80,84,82,95,69
300 DATA 78,86,1,244,7,87,73,78,95,77
310 DATA 65,78,2,186,3,71,68,50,2,38
320 DATA 7,80,84,82,86,69,82,36,2,132
330 DATA 7,79,83,95,86,69,82,36,2,246
340 DATA 7,87,77,65,86,69,82,36,0,0
350 DATA 52,120,1,24,78,146,102,0,1,118
360 DATA 87,67,102,0,1,110,74,182,152,0
370 DATA 109,0,1,102,36,118,152,0,74,182
380 DATA 152,4,109,0,1,90,38,118,152,4
390 DATA 74,182,152,8,103,14,109,0,1,76
400 DATA 34,54,152,8,22,218,83,129,102,250
410 DATA 112,0,78,117,52,120,1,24,78,146
420 DATA 102,0,1,54,83,67,102,0,1,46
430 DATA 126,2,32,54,152,0,107,0,2,216
440 DATA 97,0,3,38,107,0,2,208,112,9
450 DATA 114,50,118,255,69,250,0,10,78,67
460 DATA 32,1,96,0,2,190,34,48,16,0
470 DATA 112,0,78,117,50,48,16,0,112,0
480 DATA 78,117,52,120,1,24,78,146,102,0
490 DATA 0,242,83,67,102,0,0,234,126,2
500 DATA 32,54,152,0,107,0,2,148,97,0
510 DATA 2,226,107,0,2,140,44,8,112,0
520 DATA 78,65,32,70,32,60,0,0,0,128
530 DATA 2,130,255,0,255,255,12,130,49,0
540 DATA 48,52,101,0,2,110,112,9,114,100
550 DATA 118,255,69,250,255,176,78,67,72,193
560 DATA 32,1,96,0,2,90,32,13,144,139
570 DATA 230,136,102,0,0,156,112,16,18,60
580 DATA 0,255,20,60,0,255,78,65,126,6
590 DATA 66,128,16,1,96,0,2,62,32,13
600 DATA 144,139,230,136,102,0,0,124,112,0
610 DATA 78,65,32,8,126,6,96,0,2,40
620 DATA 0,0,0,0,0,0,0,0,58,60
630 DATA 2,0,124,0,96,22,58,60,1,0
640 DATA 124,2,96,14,58,60,0,0,124,4
650 DATA 96,6,58,60,0,0,124,6,52,120
660 DATA 1,24,78,146,102,66,83,67,102,60
670 DATA 126,2,32,54,152,0,107,0,1,232
680 DATA 97,0,2,54,107,0,1,224,118,255
690 DATA 66,130,45,73,0,88,67,250,255,178
700 DATA 112,108,78,67,34,110,0,88,74,64
710 DATA 103,4,48,5,96,8,69,250,255,158
720 DATA 48,50,104,0,72,192,96,0,1,182
730 DATA 112,241,78,117,112,250,78,117,52,120
740 DATA 1,24,78,146,102,0,255,242,83,67
750 DATA 102,0,255,234,126,2,32,54,152,0
760 DATA 107,0,1,148,97,0,1,226,107,0
770 DATA 1,140,118,255,45,73,0,88,112,112
780 DATA 78,67,34,110,0,88,74,128,102,6
790 DATA 112,1,96,0,1,116,112,0,96,0
800 DATA 1,110,52,120,1,24,78,146,102,0
810 DATA 255,178,83,67,102,0,255,170,126,2
820 DATA 32,54,152,0,107,0,1,84,97,0
830 DATA 1,162,107,0,1,76,118,255,45,73
840 DATA 0,88,112,112,78,67,74,128,103,6
850 DATA 34,110,0,88,96,196,32,9,34,110
860 DATA 0,88,74,128,111,186,96,178,52,120
870 DATA 1,24,78,146,102,0,255,112,83,67
880 DATA 102,0,255,104,32,54,152,0,107,66
890 DATA 97,0,1,100,107,60,45,73,0,88
900 DATA 118,255,112,112,78,67,34,110,0,88
910 DATA 74,128,102,42,47,1,45,73,0,88
920 DATA 114,2,48,120,1,26,78,144,34,31
930 DATA 34,110,0,88,85,137,45,73,0,88
940 DATA 61,188,0,4,152,0,45,129,152,2
950 DATA 120,1,112,0,78,117,34,110,0,88
960 DATA 84,137,45,73,0,88,66,118,152,0
970 DATA 96,234,114,6,48,120,1,26,78,144
980 DATA 34,110,0,88,93,137,112,0,78,65
990 DATA 61,188,0,4,152,0,45,130,152,2
1000 DATA 120,1,45,73,0,88,112,0,78,117
1010 DATA 52,120,1,24,78,146,102,0,254,226
1020 DATA 83,67,102,0,254,218,126,2,32,54
1030 DATA 152,0,107,0,0,132,97,0,0,210
1040 DATA 107,124,118,255,45,73,0,88,112,112
1050 DATA 78,67,34,110,0,88,74,128,102,30
1060 DATA 66,65,66,66,54,60,255,255,112,111
1070 DATA 78,67,112,1,34,41,1,40,34,110
1080 DATA 0,88,12,129,80,84,82,50,103,76
1090 DATA 112,0,96,0,0,72,52,120,1,24
```

```
1100 DATA 78,146,102,0,254,140,83,67,102,0
1110 DATA 254,132,32,54,152,0,107,0,255,94
1120 DATA 97,0,0,126,107,0,255,86,45,73
1130 DATA 0,88,118,255,112,112,78,67,74,128
1140 DATA 102,0,255,70,179,252,0,0,0,0
1150 DATA 103,0,255,60,34,41,0,0,34,110
1160 DATA 0,88,96,0,255,6,45,73,0,88
1170 DATA 50,0,36,0,103,28,50,60,8,31
1180 DATA 208,128,105,20,83,65,36,0,118,16
1190 DATA 32,2,231,160,105,4,146,67,36,0
1200 DATA 226,67,102,242,72,231,96,0,74,135
1210 DATA 103,18,34,7,48,120,1,26,78,144
1220 DATA 34,110,0,88,146,199,45,73,0,88
1230 DATA 76,223,0,6,35,130,232,2,51,129
1240 DATA 232,0,120,2,112,0,78,117,74,128
1250 DATA 107,170,192,252,0,40,208,174,0,48
1260 DATA 176,174,0,52,108,0,253,236,32,118
1270 DATA 8,0,74,182,8,0,107,0,253,224
1280 DATA 67,250,0,8
```

We have printed the BASIC listing in a larger font than the following assembler listing, as we think that most of our readers will type in the few BASIC lines and take the assembler mainly as a reference.

```
* Display Extensions V2.01, last update 21/02/03
* QLiberator directive $$asmb=flp1_display_cde,0,10
*
* Brief description and syntax of basic extensions:
*
* ADDRESS returns base address for given screen channel
* LET screen_base_address = ADDRESS(#channel)
*
* BYTES returns number of bytes per line for given window channel
* LET display_width = BYTES(#channel)
*
* DMODE returns current screen display mode (0 or 8, for example)
* LET screen_mode = DMODE(#channel)
*
* SYS_VAR returns base address of system variables area
* LET variables = SYS_VAR
*
* The FLIM_n functions use iop.flim trap to find window limits. For
* primary channels, this is screen size, for secondaries this is
* the outline limits
*
* FLIM_X and FLIM_Y return origin of largest window channel
* LET x = FLIM_X(#channel)
* LET y = FLIM_Y(#channel)
* FLIM_W and FLIM_H return limits of width and height respectively
* LET wide = FLIM_W(#channel)
* LET high = FLIM_H(#channel)

* MOVEMEM shifts a given number of bytes between two addresses,
* always from the first address to the second address, so
* overwriting (e.g. to clear an area of memory) is possible
* MOVEMEM from_address,to_address,bytes

* The new extensions in v2.00 are as follows
* (assistance from Marcel Kilgus gratefully acknowledged):
* LET v$  = OS_VER$            returns operating system version n.nn
* LET v$  = PTRVER$(#channel) returns pointer version n.nn
* LET env = PTR_ENV(#channel) returns 0 if no PE, 1 if PE present
* LET wm  = WIN_MAN(#channel) returns 0 if no WMAN, 1 if WMAN pres.
* LET gd  = GD2(#channel)     returns 0 if no GD2, 1 if GD2 present

* Added in v2.01 with information from Rich Mellor:
* LET v$  = WMAVER$(#channel)  returns Window Manager version nr

* link into basic as extensions
        lea     exts,a1         ;point to list of extensions
        move.w  $110,a2         ;bp.init
        jmp     (a2)            ;link extensions

* list of extension names and definitions
exts    dc.w    1               ;just one procedure
        dc.w    movemem-*       ;location
        dc.b    7,'MOVEMEM'     ;name
        dc.w    0               ;end of procedures list

        dc.w    14              ;14 functions
        dc.w    address-*       ;location of ADDRESS
        dc.b    7,'ADDRESS'     ;name of function
        dc.w    bytes-*         ;location of BYTES
        dc.b    5,'BYTES'       ;name of function
        dc.w    dmode-*         ;location of DMODE
        dc.b    5,'DMODE'       ;name of function
        dc.w    sys_var-*       ;location of SYS_VAR
        dc.b    7,'SYS_VAR'     ;name of function
        dc.w    flim_x-*        ;location of FLIM_X
        dc.b    6,'FLIM_X',0    ;name of function
        dc.w    flim_y-*        ;location of FLIM_Y
        dc.b    6,'FLIM_Y',0    ;name of function
        dc.w    flim_w-*        ;location of FLIM_W
        dc.b    6,'FLIM_W',0    ;name of function
        dc.w    flim_h-*        ;location of FLIM_H
        dc.b    6,'FLIM_H',0    ;name of function
        dc.w    ptr_env-*       ;location of PTR_ENV
        dc.b    7,'PTR_ENV'     ;name of function
        dc.w    win_man-*       ;location of WIN_MAN
        dc.b    7,'WIN_MAN'     ;name of function
        dc.w    gd2-*           ;location of GD2
        dc.b    3,'GD2'         ;name of function
        dc.w    ptr_ver-*       ;location of PTRVER$
        dc.b    7,'PTRVER$'     ;name of function
        dc.w    os-*            ;location of OS_VER$
        dc.b    7,'OS_VER$'     ;name of function
        dc.w    wma_ver-*       ;location of WMAVER$
        dc.b    7,'WMAVER$'     ;anme of function
        dc.w    0               ;end of function definitions
list

* code for the MOVEMEM procedure
* MOVEMEM from_address,to_address,bytes
movemem move.w  $118,a2         ;ca.gtlin to fetch long integers
        jsr     (a2)            ;try to fetch 3 long integers
        bne     return          ;oops
        subq.w  #3,d3           ;3 parameters?
        bne     badparam        ;oops
*check parameter values that addresses and bytes are positive
        tst.l   0(a6,a1.l)      ;source address
        blt     badparam        ;invalid negative address?
        move.l  0(a6,a1.l),a2   ;source address into a2
        tst.l   4(a6,a1.l)      ;destination address
        blt     badparam        ;invalid negative address
        move.l  4(a6,a1.l),a3   ;destination address into a3
        tst.l   8(a6,a1.l)      ;number of bytes
        beq.s   mvedone         ;zero bytes, don't do anything
        blt     badparam        ;invalid negative number of bytes
        move.l  8(a6,a1.l),d1   ;number of bytes to be moved
movelop move.b  (a2)+,(a3)+     ;shift a byte
        subq.l  #1,d1           ;count the bytes
        bne.s   movelop         ;keep going until all done
mvedone moveq   #0,d0           ;no error report
        rts                     ;return to basic

* LET adr = ADDRESS(#channel)
* returns base address for given screen channel
address move.w  $118,a2         ;ca.gtlin
        jsr     (a2)            ;fetch 1 parameter
        bne     return          ;oops...
        subq.w  #1,d3           ;1 parameter?
        bne     badparam        ;oops...
* fine, we have one parameter on the stack, so for any
* further errors, we can return a suitable negative number
* instead of an error report
* all the functions use d7 to indicate how many bytes more are
* required on top of any parameters already on the RI stack (the
```

Think your own thoughts.
Q60. The Super QL.



Features

- ✗ Q60/60 & Q60/66: 68060 CPU, 60/66 MHz, MMU, FPU
- ✗ Q60/80: 68LC060 CPU, 80 MHz, MMU
- ✗ 68060 superscalar architecture, dual execution units
- ✗ Up to 160 BogoMIPS performance for QDOS+SMSQ/E
- ✗ 16 to 128 MB RAM, PS/2 module sockets
- ✗ 256 kB ROM (mainboard supports up to 1024 kB)
- ✗ Highspeed 32 bit graphics + original QL hardware modes
- ✗ Up to 65536 colours at 1024 x 512 pixel resolution
- ✗ Multisync monitor output (15 pin HD connector)
- ✗ PC Keyboard interface (DIN)
- ✗ 20 kHz Stereo sound
- ✗ Battery buffered clock, 2 KB nonvolatile RAM
- ✗ Controller for 2 floppies and 2 IDE harddisks or CDROM
- ✗ 2 Serial ports with 115200 Baud, Parallel port (on I/O card supplied with mainboard)
- ✗ Hardware extension slot supports ISA cards
- ✗ Fits directly into AT Minitower or other standard case
- ✗ +5V / +12V power supply
- ✗ No tinkering, no parts from original QL needed
- ✗ Mainboard size 8.2 x 6.3 inch
- ✗ Can boot in a few seconds, directly from ROM
- ✗ Runs three different operating systems: SMSQ/E, QDOS Classic and Q60 Linux
- ✗ New „ShoeString" Q60 Linux distribution

Prices	
Complete Systems	
Q60/60 Midi Tower*	
68060 @ 60 MHz, MMU+FPU, 64MB RAM, CD-ROM 56x, 3.5" Floppy, 20 GB Harddisk, Keyboard, 3 Button Mouse, 2 SER, 1 PAR, Stereo Sound	**£ 545.00**
Higher mainboard spec.	
Q60/66 (68060, 66 MHz)	**+ £ 139.00**
Q60/80 (68LC060, 80 MHz)	**+ £ 290.00**
Extras	
RAM	
16 MB (giving 80 MB)	**£ 17.00**
64 MB (giving 128 MB)	**£ 36.00**
I/O Card (FLP,IDE,SER,PAR)	**£ 14.00**
Operating System	
OS programmed on ROMs**	**£ 10.00**
Q60 Linux CD	**£ 15.00**
Ethernet Card 10 Mbit/s	**£ 17.00**
Stereo speakers	
3 boxes, including sub-woofer	**£ 19.00**
Preinstalled software package	
QPAC1, QPAC2, FiFi, QD, PROWESS and much more, over £100 worth	**£ 59.00**

* Fully assembled and tested! Includes support disks and manuals.
** SMSQ/E and QDOS Classic available

Shipping and handling is extra. Prices may change due to semiconductor costs or exchange rates. Please note: The Q60/80 is not available with floatingpoint coprocessor. Current SMSQ/E version supports only 16 MB out of 64 MB RAM, or 32 MB out of 80/128 MB RAM. Linux fully supports all memory configurations.

Q60

Q40

Website and technical information:

www.q40.de

Email: info@q40.de

D&D Systems

P.O. Box 5813, Ripley, Derbyshire, England DE5 9ZR
Tel. +44 (0)1773-740170, FAX +44 (0)1773-748399
After sales Tel. +44(0)1773-741164 (evenings)
Email: sales@q40.de

Financially assisted by a loan from QUANTA

Take the power back in your hands.

```
 * channel number is a 4 byte value, so the norm is 2 extra bytes
 * needed to return a 6 byte float value)
         moveq   #2,d7             ;2 extra bytes will be needed
         move.l  0(a6,a1.l),d0     ;get channel number to d0
         bmi     retfp1            ;return the negative d0 value
         bsr     chan2id           ;convert d0 channel# to ID in a0
         bmi     retfp1            ;oops, return the negative number

         moveq   #9,d0             ;prepare to use the channel

         moveq   #$32,d1     ;sd.scrb offset within def. block
         moveq   #-1,d3      ;infinite timout
         lea.l   extop_l,a2 ;the routine to be invoked by sd.extop
         trap    #3          ;use sd.extop to call the routine
 * d1 now contains the value required (screen base address)
         move.l  d1,d0       ;put required value in d0 for retfp
         bra     retfp1      ;and return it

 * this routine is invoked from sd.extop to fetch a value
 * indexed by d1 from the base of the definition block in a0
 * (Thank you Norman Dunbar for originally showing me how to use
 * sd.extop)
 extop_l        move.l    0(a0,d1.w),d1 ;fetch long value required
                moveq     #0,d0         ;ensure no error
                rts                     ;and go back to caller
 extop_w        move.w    0(a0,d1.w),d1 ;fetch word value required
                moveq     #0,d0         ;ensure no error
                rts                     ;and go back to caller

 * LET bytes_per_line = BYTES(#channel)
 * returns display width for given screen channel
 * AH/JM roms (QDOS 1.03 or earlier) will return wrong result
 * so return fixed value of 128 for these roms
 bytes   move.w  $118,a2           ;ca.gtlin
         jsr     (a2)              ;fetch 1 parameter
         bne     return            ;oops
         subq.w  #1,d3             ;1 parameter?
         bne     badparam          ;oops
 * fine, we have one parameter on the stack, so for any further errors
 * we can return a suitable negative number instead of an error report

 * all the functions use d7 to indicate how many bytes more are required
 * on top of any parameters already on the RI stack (the channel number
 * is a 4 byte value, so the norm is 2 extra bytes needed to return a
 * 6 byte float value)
         moveq   #2,d7          ;2 extra bytes wil be needed
         move.l  0(a6,a1.l),d0 ;get channel number into d0
         bmi     retfp1         ;return the negative d0 value
         bsr     chan2id        ;convert channel# in d0 to ID in A0
         bmi     retfp1         ;oops, return the negative number
 * at this point:
 * A0=channel ID    A1=RI ptr      d7=extra RI stack bytes wanted

 * Thanks to Ralf Rekoendt for this tip originally
 * test if AH or JM rom (QDOS 1.03 or lower) and if so, return 128
 * the only value supported by these rom versions (fixed line width
 * of 128 bytes wide only)
 * need to preserve channel ID in a0, which is smashed by mt.inf
         move.l  a0,d6        ;save channel id which was in a0
         moveq   #0,d0        ;call mt.inf to get system info
         trap    #1           ;call mt.inf
         move.l  d6,a0        ;return channel ID to a0
 * d2.l now contains the ASCII version number (n.nn)
 * remove the decimal point to make things easier as some
 * national ROM versions contain a letter code rather than "."
         move.l  #128,d0          ;in case it's an AH/JM rom...
         and.l   #$FF00FFFF,d2    ;remove '.'
         cmpi.l  #$31003034,d2    ;>=1.04? <"1"><nul><"0"><"4">
         bcs     retfp1           ;no, return 128

 * prepare to fetch info from definition block
         moveq   #9,d0       ;prepare to use the channel
         moveq   #$64,d1     ;offset in chan def block for sd.linel
         moveq   #-1,d3      ;infinite timeout
         lea.l   extop_w,a2 ;the routine to be called by sd.extop
         trap    #3          ;invoke sd.extop
 * d1 now contains the value required (screen base)
```

```
         ext.l   d1              ;stretch to long word
         move.l  d1,d0           ;put required value in d0 for retfp
         bra     retfp1          ;and return it

 * LET mde = DMODE
 * no parameters in the basic call; return 0 or 8 for mode number
 dmode   move.l  a5,d0           ;pointer to last parameter
         sub.l   a3,d0           ;subtract pointer to parameter 1
         lsr.l   #3,d0           ;bytes DIV 8 = number of paramtrs
         bne     badparam        ;oops, silly boy
         moveq   #$10,d0         ;mt.dmode
         move.b  #-1,d1          ;READ mode
         move.b  #-1,d2          ;READ display (incidental here)
         trap    #1
 * d1.b=display mode number
         moveq.l #6,d7      ;6 bytes needed on stack to retn float
         clr.l   d0         ;empty top 3 bytes
         move.b  d1,d0      ;mode number for conversion to fp
         bra     retfp      ;send it back to basic

 * LET sys_variables = SYS_VAR
 sys_var move.l  a5,d0      ;pointer to last parameter
         sub.l   a3,d0      ;subtract pointer to 1st parameter
         lsr.l   #3,d0      ;number of parameters (8 bytes/param)0
         bne     badparam   ;oops
         moveq.l #0,d0      ;mt.inf
         trap    #1         ;call mt.inf
 * returns d2.l = QDOS ASCII version number n.nn
 * and a0 = pointer to system variables
         move.l  a0,d0      ;system variables to d0
         moveq.l #6,d7      ;space needed
         bra     retfp      ;return it to basic

 * the iop.flim based calls follow
 * LET w=FLIM_W(#channel)          LET h=FLIM_H(#channel)
 * LET x=FLIM_X(#channel)          LET y=FLIM_Y(#channel)
 * routines need a 4 byte storage area required by iop.flim for
 * width,height,x,y
 store   ds.w    4
 flim_w  move.w  #512,d5        ;default width
         moveq.l #0,d6          ;offset of width value from "store"
         bra.s   flim           ;branch to common part of routine

 flim_h  move.w  #256,d5        ;default height
         moveq.l #2,d6          ;offset of height value from "store"
         bra.s   flim           ;branch to common part of routine

 flim_x  move.w  #0,d5          ;default x
         moveq.l #4,d6          ;offset of x value from "store"
         bra.s   flim           ;branch to common part of routine

 flim_y  move.w  #0,d5          ;default y
         moveq.l #6,d6          ;offset of y value from "store"

 * common routine, with pre-assigned values in d5+d6
 flim    move.w  $118,a2          ;ca.gtlin
         jsr     (a2)             ;fetch 1 parameter
         bne.s   return           ;oops!
         subq.w  #1,d3            ;1 parameter
         bne.s   badparam         ;oops!
 * we now have the channel number on the stack
         moveq.l #2,d7            ;2 more bytes will be needed
         move.l  0(a6,a1.l),d0    ;get channel number to d0
         bmi     retfp1           ;if negative, return as fn value
         bsr     chan2id          ;channel # to ID in a0
         bmi     retfp1           ;oops,return the -ve number as
                                  ;fn value
         moveq   #-1,d3           ;timeout
         clr.l   d2               ;has to be zero
         move.l  a1,$58(a6)       ;store RI stack pointer
         lea.l   store,a1         ;point to 8 byte area
         moveq   #$6C,d0          ;iop.flim
         trap    #3
         move.l  $58(a6),a1       ;recover RI stack pointer
         tst.w   d0               ;was there any error?
         beq.s   noerr            ;no, fetch value from store to
return
```

```
* since iop.flim appears not to exist, we'll assume a 512x256 display
        move.w  d5,d0           ;default value for this function
        bra.s   extnd           ;and return that default value
* fetch values for return
noerr   lea.l   store,a2        ;point to four word store area
        move.w  0(a2,d6.1),d0   ;fetch value
extnd   ext.l   d0              ;convert to long word
        bra     retfp1          ;return as floating point value
* utility routines shared by functions
badparam moveq  #-15,d0     ;err.bp
return  rts                 ;go back to basic with d0 error report
notopen moveq   #-6,d0      ;err.no
        rts

* LET pe = PTR_ENV(#channel)
ptr_env move.w  $118,a2         ;ca.gtlin
        jsr     (a2)            ;fetch 1 parameter
        bne     return          ;oops
        subq.w  #1,d3           ;1 parameter?
        bne     badparam        ;oops
* we have channel number on stack
        moveq.l #2,d7           ;2 extra bytes will be needed
        move.l  0(a6,a1.1),d0   ;channel number
        bmi     retfp1          ;if negative, return as fn value
        bsr     chan2id         ;convert chan# in d0 to ID in a0
        bmi     retfp1          ;if -ve return as fn value
        moveq.l #-1,d3          ;infinite timeout
        move.l  a1,$58(a6)      ;store RI stack pointer
        moveq.l #$70,d0         ;iop.pinf (Get Pointer Info) trap
        trap    #3
        move.l  $58(a6),a1      ;recover RI stack pointer
        tst.l   d0              ;is there a pointer interface?
        bne.s   no              ;no there isn't
yes     moveq.l #1,d0           ;1=yes, present
        bra     retfp1          ;return 1 as function value
no      moveq.l #0,d0           ;0=no, not present
        bra     retfp1          ;return 0 as function value

* LET wm = WIN_MAN(#channel)
win_man move.w  $118,a2         ;ca.gtlin
        jsr     (a2)            ;fetch 1 parameter
        bne     return          ;oops
        subq.w  #1,d3           ;1 parameter?
        bne     badparam        ;oops
* we have channel number on stack
        moveq.l #2,d7           ;2 extra bytes will be needed
        move.l  0(a6,a1.1),d0   ;channel number
        bmi     retfp1          ;if negative, return as fn value
        bsr     chan2id         ;convert chan# in d0 to ID in a0
        bmi     retfp1          ;if -ve return as fn value
        moveq.l #-1,d3          ;infinite timeout
        move.l  a1,$58(a6)      ;store RI stack pointer
        moveq.l #$70,d0         ;iop.pinf (Get Pointer Info) trap
        trap    #3
        tst.l   d0              ;is there pointer interface?
        beq.s   ptryes          ;yes there is
* there's no pointer interface, so there can't be a window
* manager test!
        move.l  $58(a6),a1      ;recover RI stack pointer
        bra.s   no              ;no pointer interface, can't be
                                ;window manager
ptryes  move.l  a1,d0           ;window manager vector in a1
        move.l  $58(a6),a1      ;restore RI stack pointer
        tst.l   d0              ;if vector=0, no wman
        ble.s   no              ;no there isn't
        bra.s   yes             ;yes there is

* LET v$ = PTRVER$(#channel) returns pointer version n.nn
ptr_ver move.w  $118,a2         ;ca.gtlin
        jsr     (a2)            ;fetch 1 parameter
        bne     return          ;oops
        subq.w  #1,d3           ;1 parameter?
        bne     badparam        ;oops
* we have channel number on stack, so can return values
        move.l  0(a6,a1.1),d0   ;channel number
        bmi.s   retnul          ;if negative, return "" as
                                ;function value
        bsr     chan2id         ;convert channel# in d0 to ID in a0
        bmi.s   retnul          ;if -ve return "" as function value
        move.l  a1,$58(a6)      ;store RI stack pointer in bv.rip
* we are now going to call iop.pinf to check if PE available
        moveq.l #-1,d3          ;infinite timeout
        moveq.l #$70,d0         ;iop.pinf (Get Pointer Info) trap
        trap    #3              ;returns version number in d1.1

        move.l  $58(a6),a1          ;recover RI stack pointer
        tst.l   d0                  ;is there a pointer interface?
        bne.s   retnul              ;oops, no there isn't

* need 2 extra bytes on RI stack for n.nn string return
retvstr move.l  d1,-(a7)            ;stack version number temporarily
        move.l  a1,$58(a6)          ;store RI stack pointer
        moveq.l #2,d1               ;2 extra bytes needed on RI stack
        move.w  $11A,a0             ;bv.chrix vector
        jsr     (a0)                ;get 2 extra bytes
        move.l  (a7)+,d1            ;recover version number
        move.l  $58(a6),a1          ;recover RI stack pointer
        subq.l  #2,a1               ;point to length word
        move.l  a1,$58(a6)          ;store RI stack pointer in bv.rip
        move.w  #4,0(a6,a1.1)       ;4 digit string to return
        move.l  d1,2(a6,a1.1)       ;n.nn pointer version number
retstr  moveq   #1,d4               ;string result
        moveq   #0,d0               ;no error
        rts                         ;back to basic
retnul  move.l  $58(a6),a1          ;recover RI stack pointer
        addq.l  #2,a1               ;nul string needs 2 bytes, not 4
        move.l  a1,$58(a6)          ;store in bv.rip
        clr.w   0(a6,a1.1)          ;string length 0
        bra.s   retstr

* LET v$ = OS_VER$   returns operating systm version number
os      moveq   #6,d1               ;6 bytes needed to return version$
        move.w  $11A,a0             ;bv.chrix vector
        jsr     (a0)                ;get the 6 extra bytes
        move.l  $58(a6),a1          ;recover RI stack pointer
        subq.l  #6,a1               ;point to string length word
        moveq   #0,d0               ;mt.inf
        trap    #1                  ;get system info
        move.w  #4,0(a6,a1.1)       ;4 character string to return
        move.l  d2,2(a6,a1.1)       ;QDOS version number in d2.1
        moveq   #1,d4               ;string result
        move.l  a1,$58(a6)          ;store RI stack pointer
        moveq   #0,d0               ;no error
        rts

* LET gd = GD2(#channel)  returns 1 if GD2 present, 0 if not
gd2     move.w  $118,a2             ;ca.gtlin
        jsr     (a2)                ;fetch 1 parameter
        bne     return              ;oops
        subq.w  #1,d3               ;1 parameter?
        bne     badparam            ;oops
* we have channel number on stack
        moveq.l #2,d7               ;2 extra bytes will be needed
        move.l  0(a6,a1.1),d0       ;channel number
        bmi     retfp1              ;if negative, return as fn value
        bsr     chan2id             ;convert chan# in d0 to ID in a0
        bmi.s   retfp1              ;if -ve return as fn value
        moveq.l #-1,d3              ;infinite timeout
        move.l  a1,$58(a6)          ;store RI stack pointer
        moveq.l #$70,d0             ;iop.pinf (Get Pointer Info) trap
        trap    #3
        move.l  $58(a6),a1          ;recover RI stack pointer
        tst.l   d0                  ;is there a pointer interface?
        bne.s   nogd2               ;no there isn't, return 0
        clr.w   d1                  ;don't set anything, only get
        clr.w   d2                  ;address of PE linkage block
        move.w  #-1,d3              ;infinite timeout
        moveq   #$6F,d0             ;iop.slnk
        trap    #3                  ;call iop.slnk
        moveq   #1,d0               ;assume GD2 present
        move.l  $128(a1),d1         ;check GD2 flag location
        move.l  $58(a6),a1          ;recover RI stack pointer again
        cmp.l   #"PTR2",d1          ;"PTR2"=value for GD2
        beq.s   retfp1              ;yes, GD2 present, return 1
```

```
nogd2   moveq   #0,d0           ;no GD2 present
        bra     retfpl          ;return 0

* LET v$ = WMAVER$(#channel) returns wman version n.nn
wma_ver move.w  $118,a2         ;ca.gtlin
        jsr     (a2)            ;fetch 1 parameter
        bne     return          ;oops
        subq.w  #1,d3           ;1 parameter?
        bne     badparam        ;oops
* we have channel number on stack, so can return values
        move.l  0(a6,a1.l),d0   ;channel number
        bmi     retnul          ;if negative, return "" as
                                ;function value
        bsr     chan2id         ;convert chan# in d0 to ID in a0
        bmi     retnul          ;if -ve return "" as fn value
        move.l  a1,$58(a6)      ;store RI stack pointer in bv.rip
* we are now going to call iop.pinf to check if PE available
        moveq.l #-1,d3          ;infinite timeout
        moveq.l #$70,d0         ;iop.pinf (Get Pointer Info) trap
        trap    #3              ;returns version number in d1.l

        tst.l   d0              ;is there a pointer interface?
        bne     retnul          ;oops, no there isn't
        cmpa.l  #0,a1           ;is there a window man. vector?
        beq     retnul          ;oops, no there isn't
        move.l  0(a1),d1        ;get long word version number
                                ;at wman vector
        move.l  $58(a6),a1      ;recover RI stack pointer
        bra     retvstr         ;return the version nr in d1.L

* convert and return long word in d0 into a floating point
* number and return (used with permission of Simon N. Goodwin).
* d7.w contains no. of extra bytes required on RI stack over
* and above what's on the RI stack parameters
retfpl  move.l  a1,$58(a6)    ;store RI stack pointer if relevant
retfp   move.w  d0,d1         ;d1 will be exponent
        move.l  d0,d2         ;d2 will be mantissa
        beq.s   normalised    ;zero is a trivial case
        move.w  #2079,d1      ;first guess at exponent
        add.l   d0,d0         ;already normalised?
        bvs.s   normalised
        subq.w  #1,d1           ;no, halve exponent weight
        move.l  d0,d2           ;double mantissa to match
```

```
           moveq  #16,d3          ;try a 16 bit shift
normalise  move.l d2,d0           ;take copy of mantissa
           asl.l  d3,d0           ;shift mantissa d3 places
           bvs.s  too_far         ;overflow - must shift less
           sub.w  d3,d1           ;correct exponent for shift
           move.l d0,d2           ;new mantissa is more normal
too_far    asr.w  #1,d3           ;halve shift distance
           bne.s  normalise       ;try shift of 8,4,2 and 1
* is extra space required on RI stack? d7 holds number of bytes
* needed over and above existing parameter space need
* to store anything important not preserved by bv.chrix call
normalised movem.l d1-d2,-(a7)  ;store exponent/mantissa
           tst.l  d7              ;if non-zero, create more space
           beq.s  enough          ;no more needed
           move.l d7,d1           ;no. of extra bytes needed
           move.w $11A,a0         ;bv.chrix vector
           jsr    (a0)            ;do it
           move.l $58(a6),a1      ;recover ri pointer
           sub.w  d7,a1           ;adjust by no. of bytes
           move.l a1,$58(a6)      ;additional bytes pointed to now
enough     movem.l (a7)+,d1-d2    ;recover exponent/mantissa
           move.l d2,2(a1,a6.l)   ;stack mantissa
           move.w d1,0(a1,a6.l)   ;stack exponent
           moveq  #2,d4           ;signal FP result
           moveq  #0,d0           ;no errors PLEASE!
           rts

* subroutine to convert channel # in d0 to 4-byte channel ID in a0
chan2id tst.l   d0              ;can't be negative
        bmi.s   retfpl          ;oops, was negative...
        mulu    #$28,d0         ;convert into channel table offset
        add.l   $30(a6),d0      ;add start of table
        cmp.l   $34(a6),d0      ;past end of table?
        bge     notopen         ;yes, channel not open
        movea.l 0(a6,d0.l),a0   ;no-get ID into a0.l
        tst.l   0(a6,d0.l)      ;is channel open?
        bmi     notopen         ;no, return err.no
        moveq   #0,d0           ;yes, return OK code
        rts                     ;and exit
```

In the next issue, we will show you a number of examples of use of these new functions.

3D Perspective Animations - Part 2

Stephen Poole

For this Issue I decided to write a real-time animation program, so I have had to resort to using some fast SMSQ/E memory-moving commands. If you don't have SMSQ/E, maybe now would be the time for you to consider getting it, as it will run your graphics-SuperBasic programs around twelve times faster than QDOS. The program is also greatly accelerated by GoldCard or SuperGoldCard, which may be bought quite cheaply second-hand if you don't already have one.

Real-time animation is not strictly possible in Basic. To get 'real-time' output speed, at first this program will painstakingly build up each frame, then store it in memory and finally repeatedly zoom in and out of the whole scene fast enough to get the impression of retinal persistance, if you set the 'speed' to 1/50th second. For cinema you need 25 images per second. We can't achieve that, but at 24 frames with a 1/10th second sustain, we get as close as we can. Hit ESCape to quit the demonstration. This program has been severely cut down to make it short enough for you to type in. The output of my original program let me 'fly' a kilometer or so around my home village, displaying fields, trees, hedges, walls, roads, railways, houses and buildings. But this demanded a huge amount of data to be carefully typed in, and as most of these details come from Ordnance Survey Maps, it is copyright information which I cannot release. That is a pity, because the program really is much more impressive when it demonstrates scenery fly-overs. Indeed, When you fly around scenery, you get the impression that it is you that is moving, whereas in this program you get the impression that it is the 'QL' that is shifting about. But there is nothing to stop you adding SELect PROCedures to draw anything you like, and then linking them

in to the FOR loop called in the PLOT routine. You can even move objects relative to each other as was done with 'plane' in Part One. Briefly, the only limit is how much spare memory you have available.
Free_Mem gives about 1.9Mb left. Included is a hidden-face algorythm which is a floating-point version of my Radix-sort, which appeared on page 8 of QL Today Volume 2, Issue 3 of Sept-October 1997. This routine sorts through the faces of the so-called 'houses'. (Formerly, this procedure did draw houses, but now it has been simplified to produce mere pixel-boxes). If you choose to add other objects, you must link in each face using the 'ct' counter, and then call the sort faces routine, as is done for the houses. By using small-scale maps you can enter your own local scenery, starting, of course, with plots of land. Remember, about 8000 faces form a limit for my JS_QL+SGC, (as will be demonstrated, I hope, in a Quanta article). For the moment I have not attempted any shading, as this considerably increases the number of calculations and the length of the program. Indeed Shading demands the use of a complicated Ray-Tracing routine which operates as follows: Starting from the eye-position, draw an imaginary line through each pixel of the screen and extend it into the virtual scenery. If it encounters an object, bounce the ray off it and keep repeating the process until it hits a light-source or goes off to infinity. If it hits a light-source, follow the ray back through the scenery to the eye, attributing to each object hit, an ever-dimming brightness value. In this way all the scenery will be shaded, and shadows will appear automatically. As this protocol scans at least 512 * 256 * 2 = 262144 positions per screen, I will leave you to guess the time taken to completion! Furthermore, for shading, it is essential to map textures onto the facettes and calculate their brightness by interpolating between the bounce-points.
SELect structures have been chosen instead of READ and DATA for each listed item, as they can then be individually accessed, the whole lot being conveniently called by FOR loops. But take note, that for the hidden-face routine to work properly in all circumstances, the faces must be roughly the same size.
Tx,ty & tz are the trajectory's eye-positions, Cx,cy & cz are the central target_point positions, which may be placed anywhere, but which, for the sake of this demonstration, are situated in the middle of the scene. The fly-around trajectory could be smoother if it was calculated mathematically. You may like to compare code and output with the '3D TEXT' article on p.32 of the Quanta Bulletin, Volume 16, Issue 7, of August 99. The two programs use basically the same methods, but this one has retained all its functionality. Fx,fy & fz and LX,ly & lz are used to automatically compute triangles relative to the reference line. M & n are the actual intersection-points on the screen, stored in the array for retrieval after sorting. And that's all there is to it! So now you know how to set up scenes as complicated as you wish. Just write a Select list of data for each category of objects that you wish to include, with an accompanying routine to draw the standard object. You should find this easier than entering long lists of anonymous coordinates into scrolling arrays, or by mouse-clicking lines to be joined up using all three perpendicular views simultaneously. Remember that some commercial programs contain thousands of standard objects and use thousands of textures. But you still have to link it all in just the same. This is how complex video-games are built up, and there is nothing to stop you from using this program to write your own, except perhaps unless you count the value of your spare-time.... Deliberately, I have not mentioned animating people or animals, as this is theoretically incredibly difficult to achieve, so complicated in fact that big studios simply film other animals (of approximating resemblance), covered with positional markers, which can then be mirrored, morphed and texture mapped to produce the required forms. For similar reasons of facility, robots are generally programmed by simply echoing the movements of skilled tradesmen, using sensitive detection devices. There is very little 'Artificial Intelligence' in all these 3D simulations, just lots of cunning. That's because you are very limited in what you can calculate serially, whereas most phenomena occur in parallel. You will probably have seen animations on TV which fly around on Mars, Venus or elsewhere. These films are made via satellites which detect the ground features of x,y and z by radar, whose coordinates can then be fed directly into a 3D-animation program (using the same basic pricipals as in this article). As promised I have corrected the distortion that originally occurred with my listing from the 1985 program. If you set 'cz' differently to 'tz' you will again get warping, so please leave

'cz' well alone. Note that SCALE is deliberately 2 vertical units. Do not modify this value as it is a simplification to drastically reduce the complexity of the code. If you wish to zoom, do it by varying the trajectory for the view. Any camera-travelling can be done by SELecting new tx,ty,tz (& cx,cy) variables using the supplied 'Trajectory' routine. So try experimenting flying around your own local landscapes. If there is anything which is not clear from these two articles, please let Jochen know and I will try to clarify things later. All of the bugs I have encountered so far have been cleared up. If you wish to go further with perspective-animation, you will have to get involved with ray-tracing, texture-mapping, morphing and other such niceties which require huge processing power, that demands many hours of number-crunching per screen. But don't forget that most commercial 3D programs also take quite some time to master and operate, (their manuals may attain 500 pages), so don't give up too easily. When I wrote my first perspective program in 1985, commercial ones were horrendously expensive, as were the plotters required to get printed output. But now you can get some good 3D House & Garden designing PC programs for 50 to 80 euros, and get output directly from your printer.

Jochen asked me to adapt this current program to work on most QL platforms, and he has supplied me with the necessary documentation to do so, and it would be interesting to hear how the owners of Q60's get on with it in terms of speed. But unfortunately this program will not, to all intents and purposes, run on a bare QL as it only has space for two screen-frames plus a small number of faces in the array, (which itself has some 12 indexes per face to enable sorting). Two frames don't make for much of an animation!

One nasty 'bug' appeared while I was adapting this listing. It transpires that if you use NEXT from inside a subroutine to get back into a FOR loop, the loop-counter can get reactivated from outside the loop, when that loop has terminated. But if you call an epilogue subroutine from outside the loop, the problem does not occur. This caused me some considerable anguish, as I had used NEXT (to RETurn) in my program since 1984 without trouble, and it was particularly difficult to trace, as the offending subroutine was deeply nested, and only fell over at the very end of the program, after I had removed the epilogue-routine to improve this article! It took many hours of head-scratching and two letters to Jochen to get to the bottom of this! For a laugh, try breaking into the playback using 'CTRL C'. Enjoy your Helicopter flying....

```
100 ::
110 REMark QLT_3D_bas. 1985_1995, v28jan2003.
120  REMark written for JS QL with SuperGoldCard.
130  REMark Needs SMSQ/E for PEEK$ & POKE$.
140  REMark Alter 'nbr' to reduce number of frames.
150  :
160  CLEAR: REServe_MEMory: PLOT: REPLAY 10: deconfigure: INK 7: QUIT
170 :
180 DEFine PROCedure REServe_MEMory
190  CLCHP: REMark CLCHP will clear out any reserved memory.
200     nbr=24: hous=25: faces=hous*6: configure
210     :
220     OPEN#1,con_32: WINDOW 512,256,0,0: BORDER 0: PAPER 1: INK 7: CLS
230     FILL 0: SCALE 2,2/-1.5,-1.5: REMark Window 'radius' is 1.
240     :
250     REMark un TO nd are array indexes:
260      Un=1: Dn=2: up=3: Bk=4: ma=5: na=6: mb=7
270      nb=8: mc=9: nc=10: md=11: nd=12
280     REMark mx is the maximum angle from the eye to a virtual point:
290      s=PI: t=s/2: u=s*2: mx=RAD(89)
300     REMark Outrageous values are here used as flags:
310      _0=-1E600: out=-1E500
320      DIM St(faces+1,nd): REMark The main coordinates array.
330 END DEFine
340 :
350 DEFine PROCedure PLOT
360  REMark Only 8 LOCals under QDOS:
370  LOCal frames,cnt,Fx,fy,fz,Fh,fc,fb
380  LOCal ct,cubes,cube,snff,sff,a$,i$
390  LOCal dmid,lx,ly,lz,lh,e,h,d1,d2,d3,d4
400  LOCal m1,m2,m3,m4, n1,n2,n3,n4,pse
410  :
420  pse=100: REMark Pause between plots.
430 FOR frames=1 TO qn
440     cnt=frames: Trajectory cnt: CLS
```

```
450     REMark Get reference_line details:
460     Fx=cx-tx: fy=cy-ty: fz=cz-tz
470     Fh=((Fx^2)+(fy^2))^.5: fc=ATAN_(fy,Fx): fb=ATAN_(fz,Fh)
480     AT 23,3: PAPER 0: INK 7: PRINT cnt
490     :
500     REMark The sort-root is the estimated average distance:
510     DIM St(faces+1,nd): St(1,1)=800
520     ct=0: FOR cubes=1 TO hous: cube=cubes: houses cube
530     sortem: i$=INKEY$(#1,pse): IF CODE(i$)=27: QUIT
540     :
550     REMark Poke the finished frame into reserved memory:
560     snff=cnt: sff=scrn(snff): a$=PEEK$(131072,32764): POKE$ sff,a$
570 END FOR frames
580 END DEFine PLOT
590 :
600 DEFine PROCedure Trajectory(traj)
610  LOCal trj: trj=traj
620  REMark Avoid using FOR variables as formal
     parameters or SELect words:
630  REMark tx,ty,tz are the eye-coordinates:
640 SELect ON trj
650  =1 : tx=-50    : ty=-900 : tz=-200
660  =2 : tx=-50    : ty=-700 : tz=-150
670  =3 : tx=-50    : ty=-500 : tz=-100
680  =4 : tx=-35    : ty=-375 : tz=-50
690  =5 : tx=35     : ty=-300 : tz=0
700  =6 : tx=165    : ty=-325 : tz=50
710  =7 : tx=200    : ty=-450 : tz=100
720  =8 : tx=200    : ty=-600 : tz=150
730  =9 : tx=150    : ty=-750 : tz=200
740  =10: tx=0      : ty=-850 : tz=250
750  =11: tx=-200   : ty=-875 : tz=300
760  =12: tx=-400   : ty=-900 : tz=350
770  =13: tx=-600   : ty=-900 : tz=400
780  =14: tx=-800   : ty=-850 : tz=450
790  =15: tx=-1000  : ty=-700 : tz=500
800  =16: tx=-1150  : ty=-450 : tz=500
810  =17: tx=-1150  : ty=-150 : tz=450
820  =18: tx=-950   : ty=135  : tz=400
830  =19: tx=-650   : ty=200  : tz=350
840  =20: tx=-300   : ty=200  : tz=300
850  =21: tx=100    : ty=200  : tz=250
860  =22: tx=500    : ty=200  : tz=200
870  =23: tx=900    : ty=200  : tz=150
880  =24: tx=1300   : ty=200  : tz=100
890  =REMAINDER : REMark oops
900 END SELect : cx=0: cy=0: cz=tz
910  REMark cx,cy,cz are the target-point
     coordinates.
920  REMark cz MUST be equal to tz.
930 END DEFine
940 :
950 DEFine PROCedure REPLAY(speed)
960  LOCal f,loop: BEEP 12345,67
970 REPeat loop
980    REMark Zoom backwards:
990    FOR f=1 TO qn
1000        a$=PEEK$(scrn(f),32764): POKE$ 131072,a$
1010        i$=INKEY$(#1,speed): IF CODE(i$)=27:
            EXIT loop
1020     END FOR f
1030     :
1040     REMark Zoom forwards:
1050     FOR f=qn TO 1 STEP -1
1060        a$=PEEK$(scrn(f),32764): POKE$ 131072,a$
1070        i$=INKEY$(#1,speed): IF CODE(i$)=27:
         EXIT loop
1080     END FOR f
1090     REMark Hit ESCape to quit
1110 END REPeat loop
1120 END DEFine
1130 :
1140 DEFine PROCedure configure
1150  LOCal f
1160  REMark Reserve enough memory for qn screens:
1170  qn=nbr: DIM scrn(qn): FOR f=1 TO qn:
      scrn(f)=ALCHP(32764)
1180 END DEFine
1190 :
1200 DEFine PROCedure deconfigure
1210  LOCal f
1220  REMark Tidy up the common heap before quitting:
1230  FOR f=qn TO 1 STEP -1: RECHP scrn(f)
1240 END DEFine
1250 :
1260 DEFine PROCedure houses(Hzs)
1270  LOCal hzz: hzz=Hzs
1280  REMark In fact 'house' now draws 'pixel-boxes':
1290 SELect ON hzz
1300  =1 : House -500, 300,25
1310  =2 : House -400, 300,23
1320  =3 : House -300, 300,21
1330  =4 : House  100, 300,13
1340  =5 : House -600, 200,27
1350  =6 : House -200, 200,18
1360  =7 : House  100, 200,13
1370  =8 : House -600, 100,27
1380  =9 : House -200, 100,18
1390  =10: House  100, 100,13
1400  =11: House -600,   0,27
1410  =12: House -400,   0,22
1420  =13: House -200,   0,18
1430  =14: House  100,   0,13
1440  =15: House -600,-100,27
1450  =16: House -300,-100,22
1460  =17: House  100,-100,13
1470  =18: House -500,-200,25
1480  =19: House -400,-200,22
1490  =20: House -200,-200,18
1500  =21: House  100,-200,13
1510  =22: House  200,-200,10
1520  =23: House  300,-200,8
1530  =24: House  400,-200,6
1540  =25: House  500,-200,4
1550  =REMAINDER : REMark oops.
1560 END SELect
1570 END DEFine houses
1580 :
1590 DEFine PROCedure House(Hx,Hy,Hz)
1600  LOCal HL,Hw,Hh,Hr
1610  REMark Use standard model for house design:
1620  REMark HL=house_length: Hw=width:
      Hh=wall_height: Hr=roof_height
1630  HL=100: Hw=100: Hh=100: Hr=100
1640  HA=Hy+Hw: HB=Hx+HL: HC=Hz+Hh: HD=Hy+(Hw/2):
      HE=Hz+Hr
1650  :
1660  REMark Link face into face_list via count 'ct':
1670  ct=ct+1: INK 0: Q Hx,Hy,Hz, Hx,HA,Hz, HB,HA,Hz,
      HB,Hy,Hz: sort: REMark base
1680  ct=ct+1: INK 2: Q Hx,HA,Hz, Hx,HA,HC, HB,HA,HC,
      HB,HA,Hz: sort: REMark back
1690  ct=ct+1: INK 2: Q Hx,Hy,Hz, Hx,Hy,HC, Hx,HA,HC,
      Hx,HA,Hz: sort: REMark west
1700  ct=ct+1: INK 2: Q HB,Hy,Hz, HB,Hy,HC, HB,HA,HC,
      HB,HA,Hz: sort: REMark East
```

```
1710  ct=ct+1: INK 2: Q Hx,Hy,Hz, Hx,Hy,HC, HB,Hy,HC, HB,Hy,Hz: sort: REMark front
1720  ct=ct+1: INK 4: Q Hx,Hy,HE, Hx,HA,HE, HB,HA,HE, HB,Hy,HE: sort: REMark roof
1730 END DEFine
1740 :
1750 DEFine PROCedure V(vx,vy,vz)
1760  REMark ( V means View_in_3D )
1770  REMark Lx,y,z,h are the Local-triangule variables:
1780  lx=vx-tx: ly=vy-ty: lz=vz-tz: lh=((lx^2)+ly^2)^.5
1790  e=ATAN_(lz,lh)-fb: h=ATAN_(ly,lx)-fc: REMark e=slope, h=bearing.
1800  REMark Keep angles within 360 degrees:
1810  IF h>s : h=h-u: GO TO 1810
1820  IF h<-s: h=h+u: GO TO 1810
1830  IF e>s : e=e-u: GO TO 1830
1840  IF e<-s: e=e+u: GO TO 1830
1850  REMark Ensure virtual summits are in visual field:
1860  IF ABS(h)> mx OR ABS(e)> mx: m=out: n=out: RETurn
1870  REMark M is the screen perspective X-point, and n the Y-point:
1880  m=TAN(h)*-1: n=TAN(e)*(((m^2)+1)^.5): REMark gotcha!
1890 END DEFine
1900 :
1910 DEFine FuNction ATAN_(g,w)
1920  REMark ATAN_ is to overcome the weaknesses of ATAN....
1930  IF w<0 THEN
1940     IF g<0 : RETurn ATAN(g/w)-PI: END IF
1950     IF g>=0: RETurn ATAN(g/w)+PI: END IF
1960  END IF
1970  IF w=0 THEN
1980     IF g<0 : RETurn -PI/2: END IF
1990     IF g>=0: RETurn  PI/2: END IF
2000  END IF
2010  IF w>0 : RETurn ATAN(g/w): END IF
2020 END DEFine
2030 :
2040 DEFine PROCedure Q(xA,yA,zA, xB,yB,zB, xC,yC,zC, xD,yD,zD)
2050  REMark Q is to draw a quadrilateral.
2060     REMark Store the perspective coordinates:
2070     V xA,yA,zA: m1=m: n1=n: d1=lh: St(ct,ma)=m1: St(ct,na)=n1
2080     V xB,yB,zB: m2=m: n2=n: d2=lh: St(ct,mb)=m2: St(ct,nb)=n2
2090     V xC,yC,zC: m3=m: n3=n: d3=lh: St(ct,mc)=m3: St(ct,nc)=n3
2100     V xD,yD,zD: m4=m: n4=n: d4=lh: St(ct,md)=m4: St(ct,nd)=n4
2110     POINT m1,n1, m2,n2, m3,n3, m4,n4: RETurn
2120     REMark m1,2,3,4, n1,2,3,4, d1,2,3,4 come from the V routine.
2130 END DEFine
2140 :
2150 DEFine PROCedure sort
2160  LOCal Ptr,Sf,Sz
2170  REMark presorting is very fast if the QL does it during input:
2180  Ptr=1: Sf=ct+1: dmid=(d1+d2+d3+d4)/4: St(Sf,Un)=dmid
2190  REPeat pre_sort
2200     Sz=Dn+(dmid>St(Ptr,Un))
2210     IF St(Ptr,Sz)=0: St(Ptr,Sz)=Sf: St(Sf,Bk)=Ptr: EXIT pre_sort
2220     Ptr=St(Ptr,Sz)
2230  END REPeat pre_sort
2240 END DEFine
2250 :
2260 DEFine PROCedure sortem
2270  LOCal n,P,nF,loop,k,so,L,R,SL,SR
2280  REMark In this code there are no user-modifiable variables:
2290   n=0: P=1: nF=ct+1
2300  REPeat loop
2310     P=ABS(P): k=St(P,Bk)
2320     :
2330     so=((k>0)-(k<0))<0
2340     IF so: P=k: NEXT loop
2350     L=St(P,Dn): R=St(P,up): SL=(L>0)-(L<0): SR=(R>0)-(R<0)
2360     :
2370     so=SL>0 AND SR=0
2380     IF so THEN
2390        IF Done(0): EXIT loop: END IF
2400        St(P,Dn)=-L: St(P,up)=_0: St(P,Bk)=-k: P=L: NEXT loop
2410     END IF
2420     :
2430     so=SL>0 AND SR>0
2440     IF so: St(P,up)=-R: P=R: NEXT loop
2450     :
```

```
2460     so=SL=0 AND SR=0
2470     IF so THEN
2480        IF Done(0): EXIT loop: END IF
2490        St(P,Dn)=_0: St(P,up)=_0: St(P,Bk)=-k: P=k: NEXT loop
2500     END IF
2510     :
2520     so=SL>0 AND SR<0
2530     IF so THEN
2540        IF Done(0)=1: EXIT loop: END IF
2550        St(P,Dn)=-L: St(P,Bk)=-k: P=L: NEXT loop
2560     END IF
2570     :
2580     so=SL=0 AND SR>0
2590     IF so: St(P,Dn)=_0: St(P,up)=-R: P=R: NEXT loop
2600     :
2610     so=SL<0 AND SR<0
2620     IF so THEN
2630        IF Done(0)=1: EXIT loop: END IF
2640        St(P,Bk)=-k: P=k: NEXT loop
2650     END IF
2660 END REPeat loop
2670 END DEFine
2680 :
2690 DEFine FuNction Done(ESC)
2700  LOCal j,x1,y1,x2,y2,x3,y3,x4,y4
2710  REMark Ignore the root-knot: ESCape from routine when all sorted:
2720  IF P<>1: n=n+1: IF n=nF-1: ESC=1
2730  :
2740  REMark Get the perspective summits:
2750  j=P-1
2760  x1=St(j,ma): y1=St(j,na): x2=St(j,mb): y2=St(j,nb)
2770  x3=St(j,mc): y3=St(j,nc): x4=St(j,md): y4=St(j,nd)
2780  :
2790  REMark Don't draw shapes that are off-limits:
2800  IF (x1=out) OR (x2=out) OR (x3=out) OR (x4=out): RETurn ESC
2810  :
2820  REMark Draw the outlined facettes:
2830  INK 4: FILL 1: LINE x1,y1 TO x2,y2 TO x3,y3 TO x4,y4 TO x1,y1: FILL 0
2840  INK 2:         LINE x1,y1 TO x2,y2 TO x3,y3 TO x4,y4 TO x1,y1
2850  RETurn ESC: REMark ESC=1 only if the sort is completed.
2860 END DEFine
2870 ::
```

BYTS OF WOOD

SAW POINTS OFFCUTS AND SNIPPETS

This issue marks the end of the seventh volume of QL Today. I am sure that, when Stuart Honeyball and Jochen Merz contacted me seven years ago to say that Bob Dyl - publisher of IQLR - was in hospital with a heart attack and IQLR had folded, they never expected that the new magazine that we talked about would go on for so long. It is a tribute to all of the hard work and long hours put in by the various contributors, many of whom have had an article in every issue, that we have survived this long and done it so well.

Throughout these six years we have had a lot of development work put into our system. Great names have arisen and disappeared again. Throughout it all both Jochen Merz and Dilwyn Jones have tirelessly travelled the path of assembling and producing this magazine. I would like to thank them for this task and hope that we can continue for a few years yet.

None of this would have been possible, of course, without the support of our readers out there, all over the world. Thank you for buying and reading the magazine.

Meanwhile on the Other Side of the Fence.....

The mainstream computer market seems to be driving itself into a frenzy of obsolecence. The pace at which it adopts a new standard and then discards it is quite frightening and it makes our own staid pace, inching towards new interfaces and O/S updates, seem much safer. I recently installed the Norton Antivirus 2003 software onto a client's machine to find that it would not run without several updates to the ubiquitous 'Windows Exploder'.

More and more laptops come without a floppy drive now and some companies do not even provide an external version.

With the price of a 650Mb blank CD-R disk now down to the same as that of a floppy disk it becomes somewhat irrelevant to provide a floppy disk drive for the PC but it does make problems for us. There is a new standard for CD re-writables called 'Mount Rainer' Many of the newer CD writers come with this enabled so maybe we should be looking into a way to incorporate this into a QL usable form.

On another tack Keith Mitchell is looking at the Compact Flash disk he has on his exceptionally small EPIA mini ITX format PC. He has the whole thing - minus a CD drive - built into a MinisQL case. The drawback is that there is no floppy controller on the board so he is using a Compact flash card in its place. This is OK but the PC will not see this as a bootable device so he is trying to overcome this. It is, of course, no problem for QPC2 which will quite happily boot from the Compact Flash Drive.

All of these options for floppy replacement on a QL rely on the presence of the Qubide so I look forward to Nasta's new IDE interface and I may resurect my MinisQL which is currently gathering dust under the desk.

A Little Speed - Maybe too Much Haste?

Last week I put together a 3.066 GHz Pentium 4 machine for one of our customers. As always I whipped out the QPC2 CD just to see how well that would do on the system. Even running from the CD, which is a lot slower than a hard drive, it booted up my full system within 10 seconds - and that is loading ProWesS and drawing the screen, buttons, everything! Performance on this kind of system is absolutely instant. I know that you have to boot into windows first to get to QPC2 but that is pretty good if you have a fast drive, a fast board and you don't allow Windoze to waste all the CPU time loading lots of things you don't need.

3.066Ghz is currently the top speed for the Pentium 4 so it is really a premium price at the moment but the 2.8Ghz is a lot cheaper. The main difference between the two processors is that the Pentium 4 3.066Ghz CPU has Hyper-Threading capability which means that it is seen to be a quasi dual processor system which greatly increases the performance of CPU intensive processes like QPC2.

Hyper-Threading will be introduced into the lower speed CPUs later in the year and there are faster chips on the way which will knock this one from its top spot and reduce the price so one of these 'super systems' should become more affordable as time goes by. You will need Windows XP Pro for this to be effective because it is only supported on this version of Windoze

The one drawback with this is that it is sometimes too fast for the mouse and there is a tendency to 'click through'. This can be annoying if you click on an item which has a sub-menu and find that you have OK'd the sub menu without even seeing it. I think that may be adjusted in QPAC II's mouse settings but I am not sure. If not then this aspect of using fast machines needs some investigation.

The Caring Side of Microsoft

For those of you who like to have a quiet titter at the cost of those people who fall for the official Microsoft line (hook and sinker included) here is an observation that has dawned upon me over the last few months.

We all know that computer programs crash and anyone who tells you that his system never crashes or locks up is either extemporising on the truth or has never found the 'on' switch. Some systems are better than others at maintaining stability but even the most highly tweaked and refined piece of software can occasionally fall upon its sword.

In our little community a swift letter/phone call/email can usually reach the author or someone who knows the program well enough to help in solving the problem but the poor PC user is usually left floundering in the dark looking at a myriad of programs and procedures running on his machine.

When Windows XP arrived on the scene many people lauded the little box that pops up after you have summarily despatched an ailing piece of code with the dreaded 'CTRL/ALT/DELETE' karate chop. This little window helpfully says ***'You have chosen to end a non responding program. Send error message to Microsoft?'***

'*Oh, Goodie!*', You think, '*Finally Bill Gates will be looking into the cause of my problems.*' Let us not suggest that here that these programs started when you bought the O/S in the first place.

For some time at work I clicked the 'do not send' option because my natural cynicism in these matters led me to think that no one would be listening anyway. Just recently, though, I clicked 'send'. Our system at work is behind a firewall and protected by Norton Antivirus and I noticed that the little box went away and there was no murmur from either of these two staunch guardians of our Internet connection. This piqued my interest so the next time a program crashed - I did not have to wait long - I looked at the error message. There it was, a long complicated dump of all of the processes that went on at the time of the crash. I clicked send and absolutely nothing happened except the box went away.

I brought this up with a network engineer friend of mine who had a similar observation. One machine he had been working on was behind a very tight firewall which let nothing in or out of the machine with asking permission and that didn't raise a cybernetic eyebrow.

So, either the Gatesian Empire has found a way of transmitting pure information over the air, or a way of tunnelling through the toughest firewall and antivirus system or the whole thing is an elaborate ruse to me you believe some one is listening when there is no one there at all.

Can you guess which explanation I plump for?

One New Years Wish Fulfilled

In my last column I wrote about the need to get some replacement for the QL keyboard membranes since they were beginning to fail at an alarming rate. I also mentioned that someone, whose name I temporarily forgot, was organising the manufacture of more membranes but I had heard very little about it since the first announcement.

Well, I am pleased to not only be able to tell you his name and the name of the company doing the job but to relate that the work seems to be progressing and the new membranes should be rolling of the production lines very soon. The person who did the organising of this was Callum Davidson and the company is called Sintech.

Callum was inspired into this by cries from other users who had no access to new membranes and egged on by Phoebus Dokus and Rich Mellor so a big well done to them. It seems that Rich Mellor's RWAP Software will be handling the distribution of these items although some of the other traders (myself included) may be holding a small stock.

It is not that I am a great supporter of the original hardware school, as you all well know, but I would be sad to see all of the black boxes disappear. I still use my JM ROM version with a 768K Trump Card to make the disks for this magazine.

Now all we need is to get the rest of the wishes off the ground - new expansion card anyone?

A Few Kind Words

One of the nice things about doing all of the subscription renewals for this magazine is reading the comments written on the forms. On the whole most are full of praise for the authors, editor and publisher who put some much effort into bring this out every two months. Some are constructive and I try to pass those on to the editor and publisher wherever possible so feel free to write in and tell us what you need. Thank you for your good wishes and support.

Stephen Poole wrote in to suggest that everyone who owns an original Black Box QL should buy one of the membranes mentioned above - if only as a spare. I am inclined to agree with him in this, especially because the people who organised them will be putting a lot of money into buying these membranes and we should all support their effort.

And, yes, I know, Chris, that black ink on a grey background is not that legible. I did expect the grey to be a lot lighter and it was so on my printer but I had the renewal forms printed by the local copy shop and they wound up a lot darker than I thought. Sorry if I strained your eyesight a bit.

A slip of the Digit

Many of you may have realised that I made a slight error on the renewal forms that were sent out with the last issue. Since we have to raise the price of the magazine for the next volume Jochen and I decided that we would offer those who renewed early renewal at the old price - as a 'thank you' for your loyalty. This worked well and I would like to offer a further thank you for your quick response.

I did, however, make a slip on the UK pricing which meant that instead of offering it at £26.00, which was last years cover price, I did it at £25.00 which was the price from three years ago. I did not realise the error until the cheques started arriving. Since I had made the offer and could think of no way of telling everyone it was wrong anyway I accepted this as my mistake and honoured the price.

The point of this little discussion is that some of you may have noticed that the great online trading emporium, Amazon, made a similar, if more disastrous, slip in pricing its IPAQ hand held computers. It was offering the £500, top of the range, model for £23.50 and the £250, low end, one for £9. I got my order in quick and actually got an order confirmation from Amazon but they hastily shut the site down and then issued a retraction and cancellation of the orders.

There are still arguments going on about whether they can actually do this. They are saying that there was no contract in place and it was just an offer to buy on their part which they can reject - but their order confirmation clearly states 'to cancel this contract....'

As the legal eagles flex their muscles it is nice to feel a bit more honest with my customers than Amazon.

Going Native on PC Hardware

Another reader has expressed views upon the concept of SMSQ/E running as a native application on PC hardware - i.e. without the presence of Windoze or another medium used as a 'host' system. This has been mentioned before by people who have a deep seated hatred of Micro$oft (heightened by Bill Gates' latest share dividend award I have no doubt). This concept is, unfortunately a non-starter. The problem is that PC hardware is not one or two motherboards and a few peripheral cards but a whole host of different processors, chipsets and peripherals. Even if you were to limit yourself to Pentium 4 processors and Intel boards there are several different chipsets and associated components that would keep you coding for years just to get the board up and running without any of the 'fancy' add-ons such as USB 1.1! Given that the bare nuts and bolts of these chipsets are hard to prise from the sticky fingers of the developers and that we would be trying to run a system based on a Motorola CPU on something completely alien, just getting a beep from the speaker would be a major achievement.

And then there is the whole mess of graphics cards and drivers, for which there are as many as for the motherboards. Given the speed of obsolescence I mentioned before I think that you can appreciate that by the time it was ready to beta test there would be no more boards available for it to run on. Just give thanks for QPC2.

SOQL

Many of you may have noticed that there have been a few emails from Jonathan Dent recently which have been sent from his QL system. This would seem to indicate that the long awaited goal of email on a QL is approaching a reality. I gather that this is fairly crude at the moment but, once the connectivity is established a proper interface and editing software could be produced from existing QL programs.

Jonathan has been asking for volunteers to beta test the current version and, judging by some of the emails on the list people are beginning to get to grips with it.

This will allow many more QL users to participate on the QL Users list which would be a very good thing wouldn't it?

A New Look for the Summer

There has been a recurring theme at workshops and in emails from Q Branch Qustomers. People have been impressed with the way that new new colour schemes can be incorporated into the BASIC windows but, say many of you, 'Where are the new programs?'

Marcel's sterling work in incorporating the colour schemes into the new WMAN code and thus into SMSQ/E is now finally bearing fruit. Fresh from falling down a mountain with two bits of wood (no, not pages from this magazine) tied to their feet, Bernd and Jochen have concocted new versions of QMenu, QD, and QSpread. It seems that their Apre Piste was hunching down over the laptops and writing code.

These are not available to the public yet and would, in any case have to have the new version of QPC2/SMSQ/E which is soon to be released but, judging from the test versions that I have, prepare to be amazed. I can truthfully say that QL programs have never looked so good.

I will give more details in the next issue because space here is tight but do try to get to see these being demonstrated at a workshop near you soon. Watch out too for the next Honourable Mention. Jochen, Bernd and Jim Hunkins could be fighting it out for the award if all this arrives at the same time!

A Computer Glossary

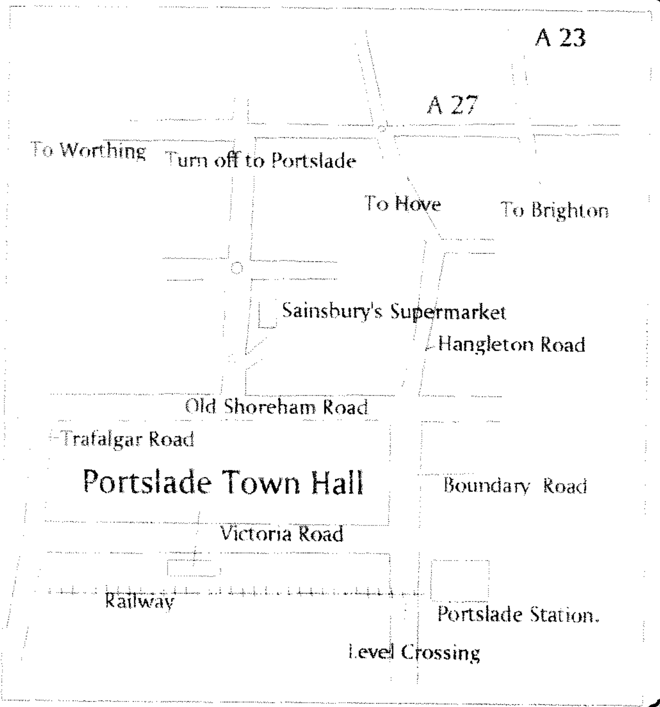
Some people may not be up to date with the latest terms used by the computer industry. Here is a list of the most prominent ones so you do not feel too uniformed when going to your local computer shop.

BIOS	To favour one system over another.
RAM	Accepted method of inserting a disk into a drive
Floppy Disk	Storage Media accidentally left on the gas fire
Hard Disk	Storage Media you don't understand
Reboot	The practice of kicking the case a second time to see if that solves the problem
Driver	The person who stays sober at a QL workshop to get everyone else home.
Zip Drive	Trying to do up your trousers in a hurry.
CD Burner	Small stove used to destroy AOL disks
Virtual Memory	Not remembering your password
Recurssion	Swearing at the computer - again!
Small Footprint PC	Marks on the tower case which show that your wife hates you sitting at the computer until three in the morning.
Digital Signature	Fingerprints on the monitor after the kids have been at it.
Serial Port 1	Drinking one glass after another
Parallel Ports	Dover and Calais
The Screen	Screensaver by Edvard Munch showing a computer user whose hard disk has crashed.
Flat Screen	What happens when you throw a monitor from a tall building.
Screensaver	Person, on whom the above monitor falls, thus cushioning its impact.
Back Up	A feeling of irritation when someone says 'I told you to keep a copy'
Sound Card	One that works.
Firewire	What happens if you throw the switch from 240 to 110 on the PSU.
Motherboard	Still living with your parents.
Megahertz	Extreme pain felt by PC users when their CPU is not as fast as everyone elses.
Update	Evening out with a relentlessly cheerful woman
Downgrade	Quality control for duck feathers
Routine	Young Australian marsupial
Sub-routine	Underwater dance performance
Software package	Christmas cardigan sent by an elderly relative
Hardware patch	Leather elbow protectors
Firmware upgrade	Silicone breast implants
Chipset Drivers	Mobile caterers
Twain Driver	Railway worker with a cleft palette
Graphics Tablets	Designer drugs
Directory Tree	Arboureal growth destined to be pulped and turned into a big list.
Smart Media	A contradiction in terms.
Compact Flash	Exhibitionist dwarf.
Icon	Overpriced Apple accessory.
I-Mac	Scots raincoat
Video Ram	Welsh porn star.
Tweeter	What Spanish birds do.
Sub-Woofer	Gay U boat comander
Serial Port 2	Some computers have more than one serial port which can be .... (To Be Continued)

The QL Show Agenda

Hove Workshop - (UK)

Quanta AGM and Workshop
Portslade Town Hall
Hove, Sussex
May 4th 2003



This year's Hove Workshop is being held later than usual because it is also going to be the venue for the QUANTA AGM. This will be our 9th show and the third one to be held in this venue. We hope to publish a list of local hotels and guesthouses in the next issue.
As usual our bevy of local ladies will be on hand to provide refreshments.
See you all there.

North American US Show 2003

Quanta and NESQLUG are pleased to announce the US QL show to be held **Saturday 17 May 2003** from 9 AM to 5 PM at the Econo Lodge at 370 Highland St., West Haven, Connecticut 06516-3522.

West Haven is on the coast adjacent to New Haven. The special rate at the Econo-Lodge is $59 (including tax!) per room per night for 1 to 4 persons if you make reservations before 17 April. Call 203 934-6611, email: **econolodge@comcast.net** or mail. Please mention "Albert rate" and include your credit card number. Continental breakfast (coffee and pastry) is included.
New Haven Tweed (HVN) is the closest airport, but the closest international airport is 50 miles away - Bradley International in Hartford, CT. The New York airports JFK and La Guardia are a little over one hour away. Newark Airport in New Jersey is not much further but requires a ride through New York City. NESQLUG will endeavor to provide rides for those arriving by air. Please contact Bill Cable, email **cable@cyberportal.net** if you need a need or can help out with a ride.
The Econo Lodge is 2 miles from the beach.
From the north, take I-95 exit 42, take right turn to Route 162 East, hotel is a half mile on the left.
Several restaurants and a shopping mall are nearby. Those who arrive by 6 PM Friday may optionally meet in the parking lot to eat together in a recommended restaurant. Nearby New Haven is the home of Yale University and contains several museums and other tourist attractions. Many other attractions are along the Connecticut coast, plus there is good and cheap public transportation to New York City. Ladies will meet at 10 AM to make plans with Dorothy Boehm to see nearby sights.
Contact Al Boehm, tel: 256 859-8051 or email **albertboehm@juno.com** for further information.

Looking forward to seeing you all again: J-M-S, QBranch and Marcel Kilgus will be there!

QL Meeting - (NL) Eindhoven

Saturday, 14th of June, 10:00 to 16:00
Pleincollege St. Joris, Roostenlaan 296